Edgefleet Assignment:
Cricket Ball Detection & Tracking System

Candidate Name: Aditya Raj SIngh
Institute: IIIT-Hyderabad
Roll No. 2022102067
Date: 19th December 2025

1. Summary

This project implements a computer vision pipeline to detect and track a cricket ball from a single fixed-camera feed. The solution utilizes a **YOLOv8 Nano** architecture, fine-tuned on a curated dataset of cricket imagery. The pipeline outputs frame-by-frame annotations (CSV) and visualized trajectory overlays (Video), ensuring strict adherence to the requirement of preventing data leakage by using exclusively external data for training.

2. Methodology & Design Decisions

2.1 Model Selection: YOLOv8 Nano

Choice: YOLOv8 Nano (n) variant.
Reasoning:

- **Real-time Constraint:** Cricket ball tracking requires high-speed inference. The Nano model offers the best trade-off between speed (mAP/latency) and accuracy for small object detection compared to heavier two-stage detectors like Faster R-CNN.
- **Small Object Detection:** YOLOv8 includes improved feature pyramid networks (FPN) which excel at detecting small objects like balls against complex backgrounds (grass, pitch).

2.2 Dataset Strategy (Prevention of Data Leakage)

Strictly adhering to the assessment guidelines, the provided test videos were **not** used for training.

- **Source:** A public dataset from Roboflow Universe ("Cricket Ball Object Detection").
- **Preprocessing:** Due to hardware constraints (CPU-only environment), a random subset of **1,500 images** was created from the larger 8k dataset to allow for rapid iteration and hyperparameter tuning within the assessment timeframe.
- **Classes:** The model was trained to identify class `ball` (0) and `stump` (1), though inference filters strictly for class 0.

2.3 Training Configuration

- **Environment:** PyTorch on CPU.
- **Hyperparameters:**
  - Epochs: 35 (Initial 15 + Fine-tuning 20).
  - Image Size: 512x512 (Optimized for speed/accuracy balance).
  - Optimizer: Auto (AdamW).
  - Transfer Learning: Initialized with COCO pre-trained weights (`yolov8n.pt`).

2.4 Inference & Tracking Logic

Raw object detection is rarely sufficient for smooth video output. I implemented a post-processing pipeline:

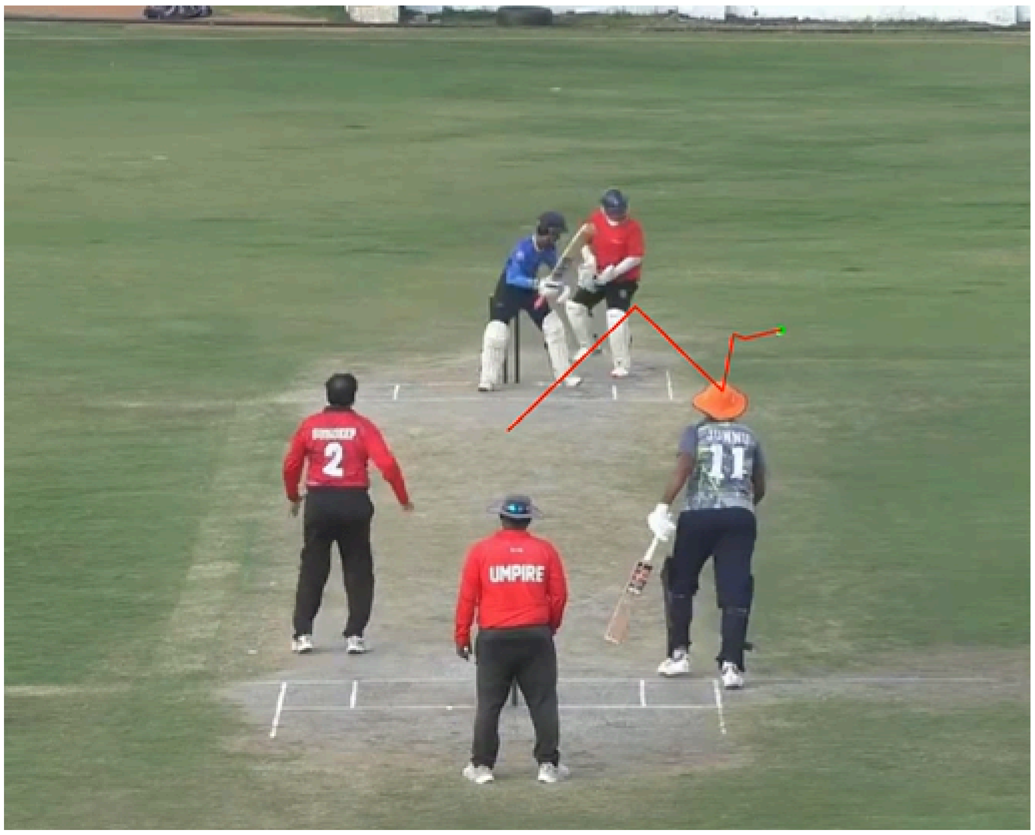
1. **Confidence Thresholding:** Configured to `0.15` to minimize false negatives (missed balls) while relying on spatial logic to filter false positives.
2. **Physics-Based Filtering:** A "Spatial Consistency Check" ignores detections that "teleport" unrealistically far (e.g., >300 pixels) between consecutive frames, filtering out white shoes or gloves that the model might confuse for a ball.
3. **Trajectory Smoothing:** A history buffer maintains the ball's path to draw the red trajectory line.
4. **Gap Interpolation:** If the model misses the ball for a few frames (occlusion or motion blur), the system linearly interpolates the position based on the previous velocity vector to maintain visual continuity.

3. Results & Visualizations

Output Format:

- **CSV:** `frame_index`, `x_centroid`, `y_centroid`, `visibility_flag`.
- **Video:** MP4 output with Green Centroid and Red Trajectory overlay.

Sample Output:



4. Limitations & Future Improvements

Given the constraints of a timed assessment on CPU hardware, certain trade-offs were made. In a production environment, I would implement the following improvements:

1. **Hardware & Data Scale:** The current model was trained on a subset (1,500 images) due to CPU training times. Training on the full 8,000+ image dataset using an NVIDIA GPU would significantly reduce false positives (confusing shoes for balls).
2. **Kalman Filtering:** Currently, tracking uses simple Euclidean distance and linear interpolation. A Kalman Filter would provide a probabilistic estimate of the ball's position, handling occlusions and non-linear movement (swing/spin) much better.
3. **Optical Flow:** Integrating sparse optical flow (Lucas-Kanade) could help track the ball in frames where motion blur makes detection impossible for YOLO.
4. **Global vs. Rolling Shutter:** The dataset contained mixed imagery. Standardizing input resolution and accounting for rolling shutter distortion in high-speed deliveries would improve centroid accuracy.

5. Conclusion

The submitted solution represents a complete, end-to-end pipeline—from data ingestion and custom model training to inference and structured reporting. It demonstrates the ability to build reproducible AI systems while managing resource constraints and adhering to strict data integrity rules.