# Cricket Bat & Ball Detection + Speed Estimation System

This project detects **cricket ball and bat**, tracks the **ball trajectory**, and estimates the **ball speed when it reaches the batter** using **YOLOv8** and **OpenCV**.

The system combines two trained models:

1. **Ball Detection Model**
2. **Bat Detection Model**

Both models are used together to accurately detect, track, and estimate the ball's speed in real-time from a video.

### Paths to the models

The YOLOv8 models for ball and bat detection are saved at:

- **Ball model:** D:\Projects\ball_detection\runs\cric_detect_train_bat_ball_sample2\weights\best.pt
- **Bat model:** D:\Projects\ball_detection\runs\cric_detect_train_bat_ball_finetune3\weights\best.pt

---

## Project Structure

```
ball_detection/
│
├── ball_dataset/                        # Dataset for training ball detection
model
├── bat_dataset/                         # Dataset for training bat detection
model
│
├── notebooks/
│   ├── Ball_model_training.ipynb        # Notebook used to train YOLO model for
ball detection
│   └── Bat_model_training.ipynb         # Notebook used to train YOLO model for
bat detection
│
├── runs/
│   ├── detect/
│   │   ├── cric_detect_train_bat_ball_sample2/   # Folder containing the trained
ball detection model
│   │   └── cric_detect_train_bat_ball_finetune3/ # Folder containing the fine-
tuned bat detection model
│
├── utils/
│   ├── tracker.py                       # Custom object tracking class (centroid
+ IoU based)
│   └── speed_estimation.py              # Functions for speed calculation and
smoothing
```

```
│
├── vedios/
│   ├── input/                          # Input video(s)
│   │   └── example_clip.mp4
│   └── output/                         # Processed videos (saved here after
running detection)
│       └── processed_clip.mp4
│
└── videos/
    └── detection_and_tracking.ipynb    # Main notebook to run detection,
tracking, and speed estimation
```

---

## Environment Setup

1. **Clone or download** the project folder `ball_detection/` to your local system.
2. Open a terminal or Anaconda prompt inside the folder.
3. Create and activate a virtual environment (recommended):

```
conda create -n cricket_env python=3.10 -y
conda activate cricket_env
```

4. **Install dependencies:**

```
pip install ultralytics opencv-python numpy
```

---

## Model Training

### 1. Ball Detection Model

- **File:** `ball_detection/notebooks/Ball_model_training.ipynb`
- **Description:** Trains a YOLOv8 model using the dataset inside `ball_dataset/`.
- **Result:** The trained weights are saved inside:

```
ball_detection/runs/detect/cric_detect_train_bat_ball_sample2/weights/best.p
t
```

---

### 2. Bat Detection Model

- **File:** `ball_detection/notebooks/Bat_model_training.ipynb`

- **Description:** Fine-tunes YOLOv8 for bat detection using `bat_dataset/`.
- **Result:** The fine-tuned model is saved inside:

```
ball_detection/runs/detect/cric_detect_train_bat_ball_finetune3/weights/best
.pt
```

---

## Components Explanation

### 1. `utils/tracker.py`

Implements **object tracking** based on:

- Centroid distance
- Intersection over Union (IoU)
- Unique IDs for consistent tracking across frames

### 2. `utils/speed_estimation.py`

Contains:

- `calculate_speed()` → Converts pixel movement to real-world speed (km/h)
- `SpeedAggregator` → Smooths speed over frames and freezes at the hit
- Utility functions for overlay and UI display

### 3. `videos/detection_and_tracking.ipynb`

Main notebook that:

- Loads both **bat and ball models**
- Detects objects frame-by-frame
- Tracks the ball trajectory
- Calculates and displays **instantaneous** and **maximum speed**
- Detects **"hit" event** (when bat and ball overlap)
- Saves processed output video to:

```
ball_detection/vedios/output/processed_clip.mp4
```

---

## How to Run the System

1. Place your cricket video inside:

```
ball_detection/vedios/input/
```

Example: `ball_detection/vedios/input/example_clip.mp4`

2. Open the notebook:

```
ball_detection/videos/detection_and_tracking.ipynb
```

3. Run all cells in order. It will:
    - Load the two trained YOLO models
    - Detect & track the ball and bat
    - Calculate ball speed frame-by-frame
    - Display results with overlays
    - Save processed output at:

```
ball_detection/vedios/output/processed_clip.mp4
```

---

## Speed Estimation Notes

The **ball speed** is estimated using:

- Distance covered between consecutive frames
- Frame rate (FPS) of the video
- Pixel-to-meter conversion factor

**Formula:** $$ Speed (m/s) = \frac{\text{Distance in pixels} / \text{pixels per meter}}{\text{Time per frame}} $$ Converted to **km/h** for display.

---

## Limitations

- Works best on **high-quality videos** (720p or above).
- If the video is **blurry** or **too dark**, detections may be inaccurate.
- Models are trained for **specific cricket angles** — performance may vary with different camera perspectives.
- Ensure FPS and camera distance are consistent for accurate speed estimation.

## Output Example

Once the system runs successfully, you'll find:

- **Detected and tracked video:** `ball_detection/vedios/output/processed_clip.mp4`
- **Speed displayed on video overlay** (in km/h)
- **Hit event** highlighted when bat and ball make contact