GAMESENSE

DYNAMIC BALL TRACKING & HIGHLIGHT GENERATION

Submitted To:-

Dr. Gaurav Singal

Department of Computer Science

Submitted By :-

Sudhanshu Kumar 2021UCD2157

PROBLEM STATEMENT?

The current method of creating highlights from cricket matches relies on human editors, which is a *time-consuming* , *resource-intensive* and often results in delayed availability of highlights.

- **Dedicated Human Editors:**
- **Time-consuming:**
- **Resource Intensive:**
- **Delayed Availability:**

OBJECTIVE

A cricket match analysis system with computer vision for ***ball tracking***, ***decision-making***, complemented by ***highlight generation.***

Cricket Match

INPUT

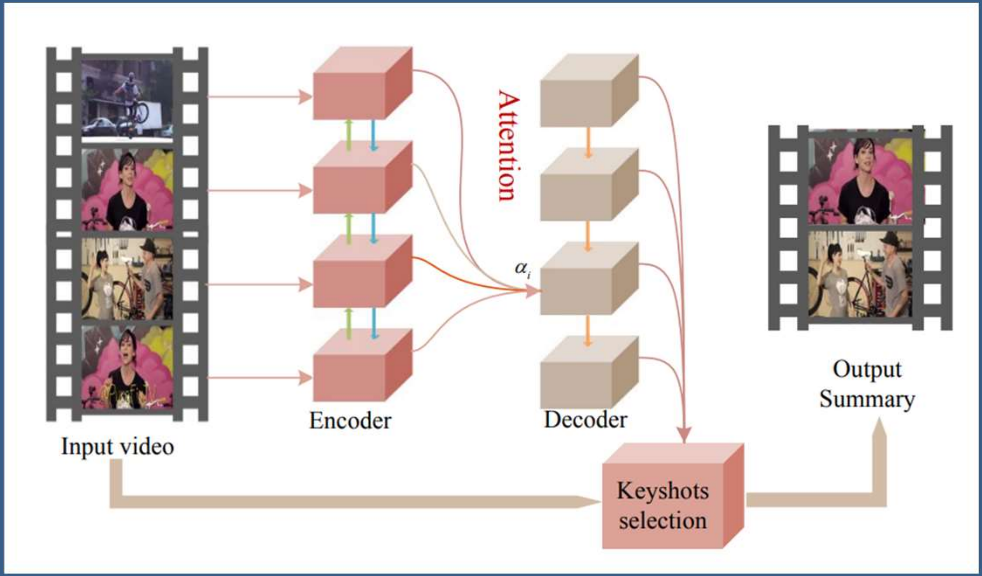
Processing

OUTPUT

METHOLOGY -1

- ❖ Low Accuracy
- ❖ Limited Contextual Understanding
- ❖ Overfitting and Generalization
- ❖ Huge Data is need for training purpose

Using Transformer

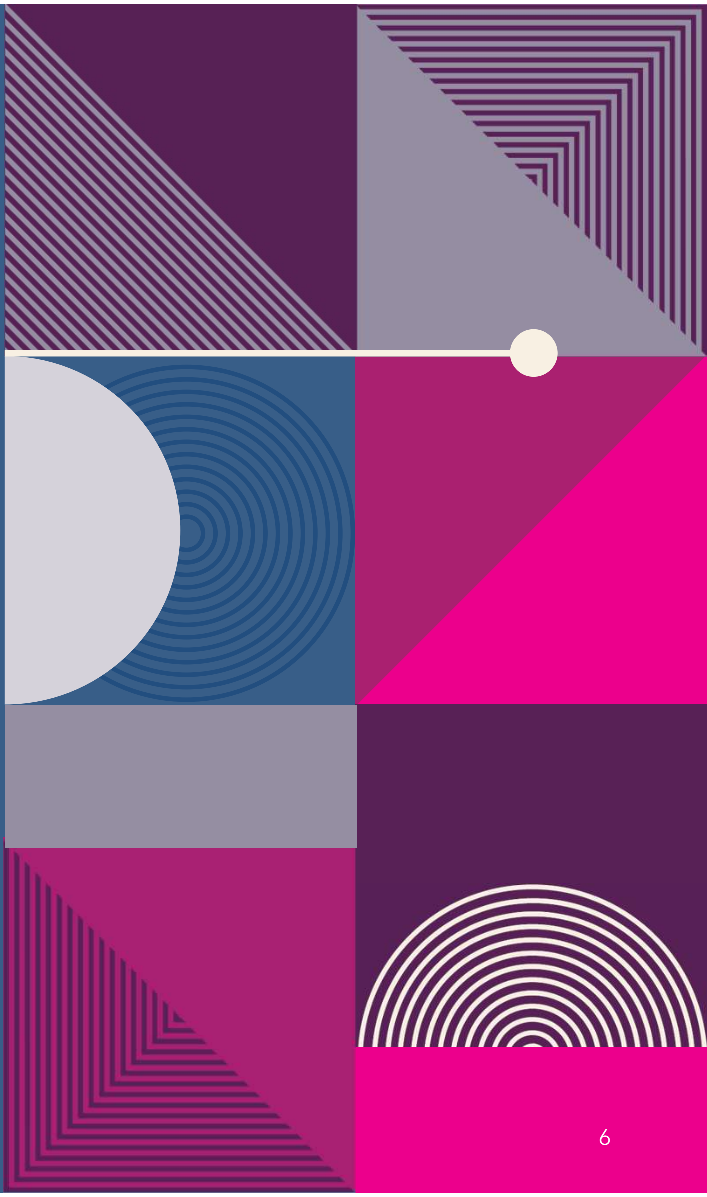


METHOLOGY - 2

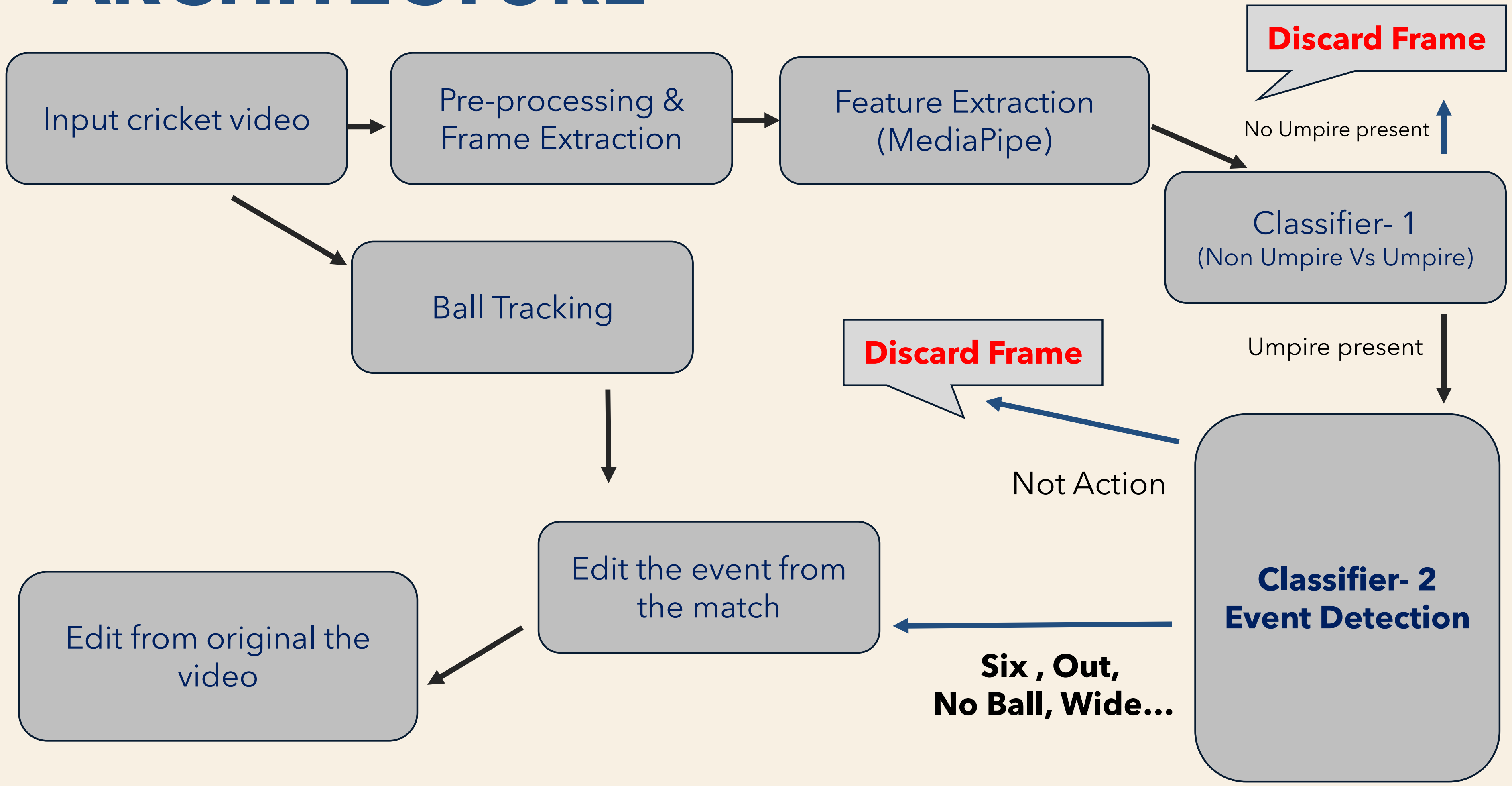
- ❖ 1.Video preprocessing
- ❖ 2. Ball Detection
- ❖ 3. Ball Tracking
- ❖ 4. Compare with Umpire
- ❖ 5. Make the decision
- ❖ 6. Make the time stamp for that shot
- ❖ 7. Make the highlight from Orignial Video

NOVELTY

- ✓ **Unique Perspective:** Utilizing tracking and mapping technology from the umpire's viewpoint offers a distinct angle for cricket highlights, enhancing viewer engagement.
- ✓ **Key Moment Prioritization:** By mapping the umpire's movements, highlight generation algorithms can prioritize critical match moments like dismissals and close calls, improving highlight reel quality.
- ✓ **Interactive Analysis:** Integration of tracking data allows for interactive viewing experiences, enabling viewers to analyze match dynamics and player performance in real-time.



ARCHITECTURE


Input cricket video
Pre-processing & Frame Extraction
Feature Extraction (MediaPipe)
Classifier- 1 (Non Umpire Vs Umpire)
No Umpire present
Discard Frame
Umpire present
Classifier- 2 Event Detection
Not Action
Discard Frame
Six , Out, No Ball, Wide...
Ball Tracking
Edit the event from the match
Edit from original the video

SOFTWARE USED

Jupyter notebook

VS code

YT Downloader

uTorrent

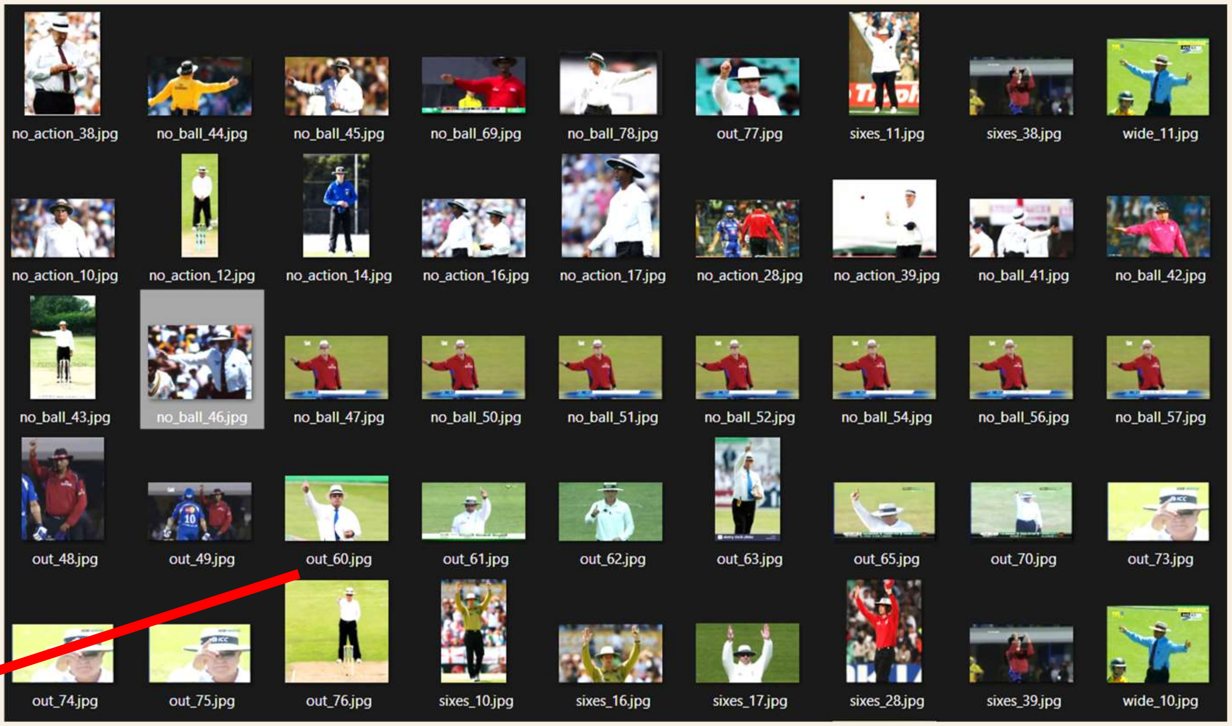
Packages used:-

- **MoviePy:** Python library for video editing.
- **FFmpeg:** Multimedia framework for decoding and encoding various media file formats.
- **TensorFlow (YOLOv7):** Implementation of YOLO object detection framework in TensorFlow.
- **Keras:** High-level neural networks API for building and training deep learning models.
- **IPython:** Toolkit for interactive computing in Python.

UMPIRE DATASETS

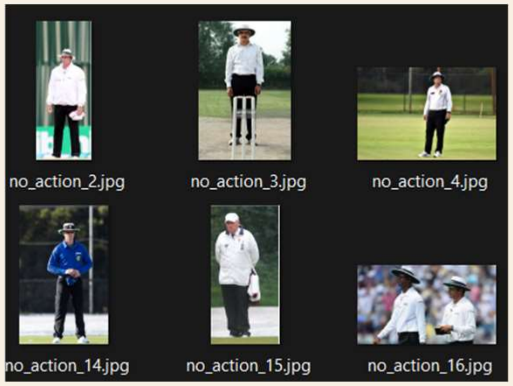
Source:- Downloaded from google images

- OUT
- NO ACTION
- NO BALL
- WIDE
- SIXES
- FOUR…

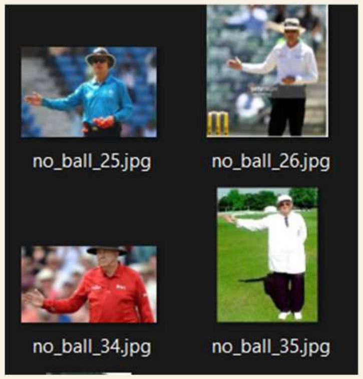


Labelled Datasets

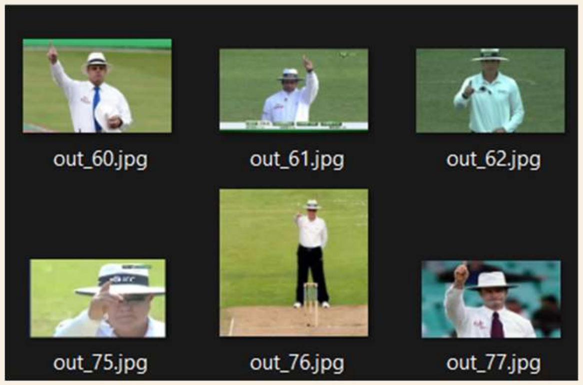
UMPIRE DATASETS



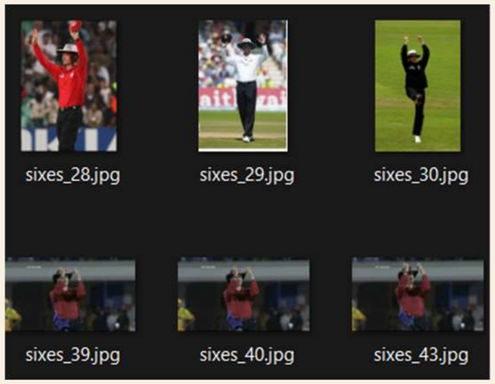
No Action (70 Instances)



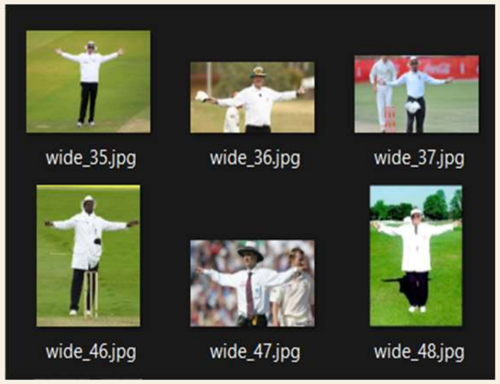
No Ball (70 Instances)



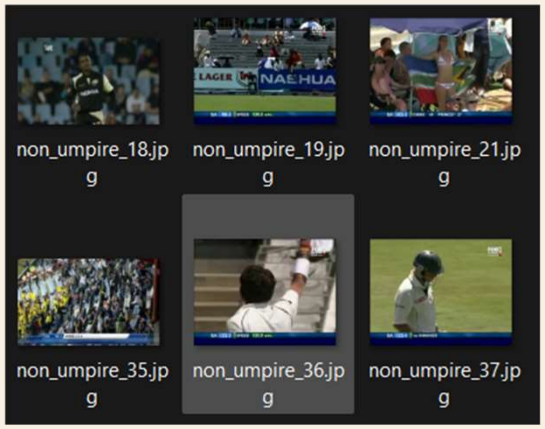
Out (75 Instances)



Sixes (70 Instances)



Wide (70 Instances)

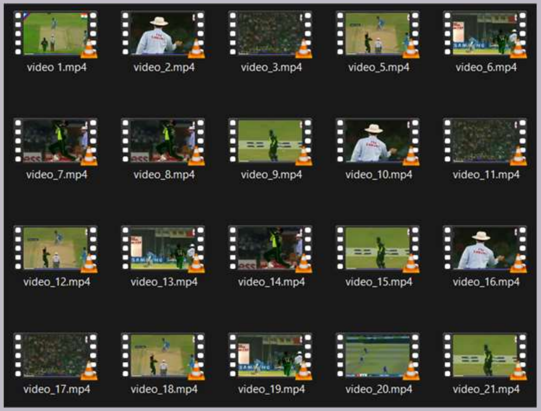


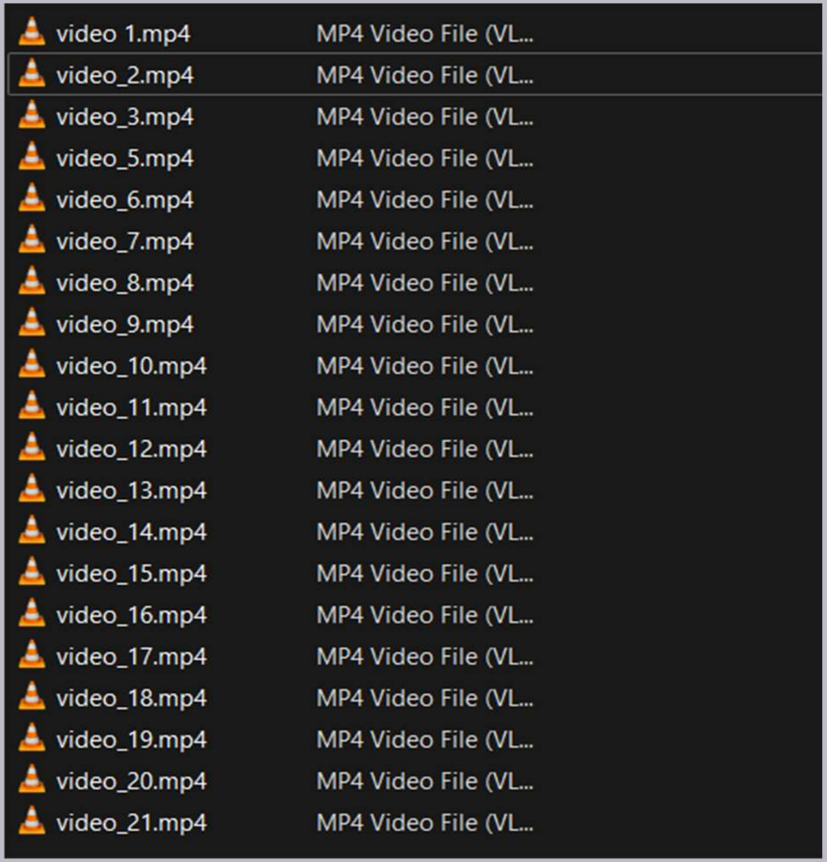
Non-umpire (150 Instances)

CRICKET VIDEO DATASETS

Source:- Downloaded from the YouTube, Google

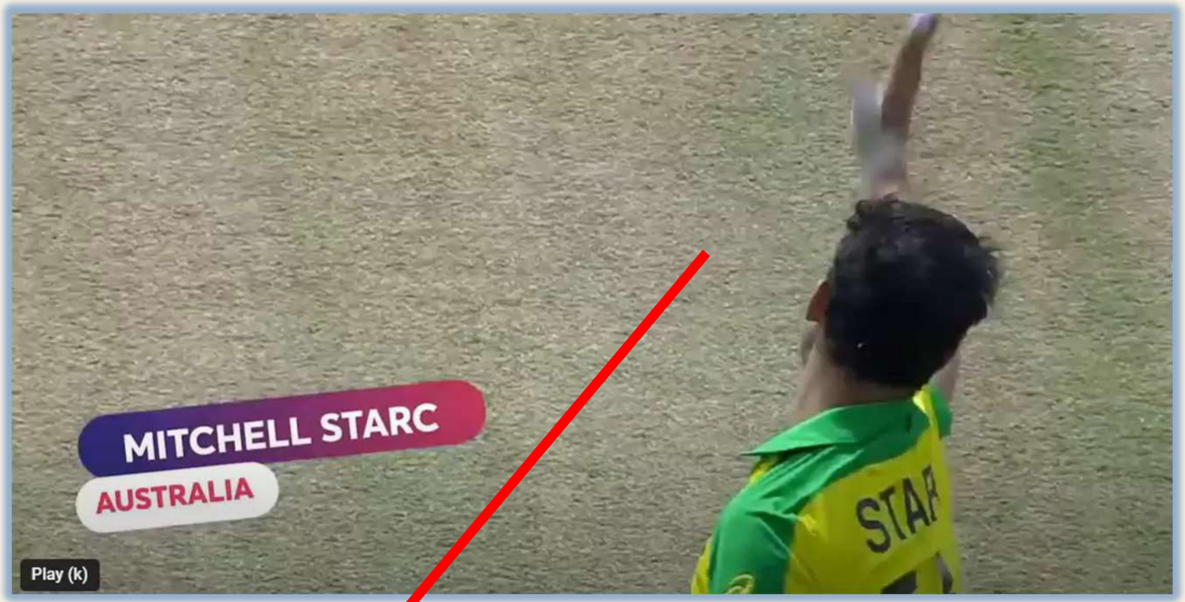
- **21 Videos**
- **MP4 Video**
- **30 FPS Video**
- **20-25 mins**





video 1.mp4	MP4 Video File (VL...
video_2.mp4	MP4 Video File (VL...
video_3.mp4	MP4 Video File (VL...
video_5.mp4	MP4 Video File (VL...
video_6.mp4	MP4 Video File (VL...
video_7.mp4	MP4 Video File (VL...
video_8.mp4	MP4 Video File (VL...
video_9.mp4	MP4 Video File (VL...
video_10.mp4	MP4 Video File (VL...
video_11.mp4	MP4 Video File (VL...
video_12.mp4	MP4 Video File (VL...
video_13.mp4	MP4 Video File (VL...
video_14.mp4	MP4 Video File (VL...
video_15.mp4	MP4 Video File (VL...
video_16.mp4	MP4 Video File (VL...
video_17.mp4	MP4 Video File (VL...
video_18.mp4	MP4 Video File (VL...
video_19.mp4	MP4 Video File (VL...
video_20.mp4	MP4 Video File (VL...
video_21.mp4	MP4 Video File (VL...

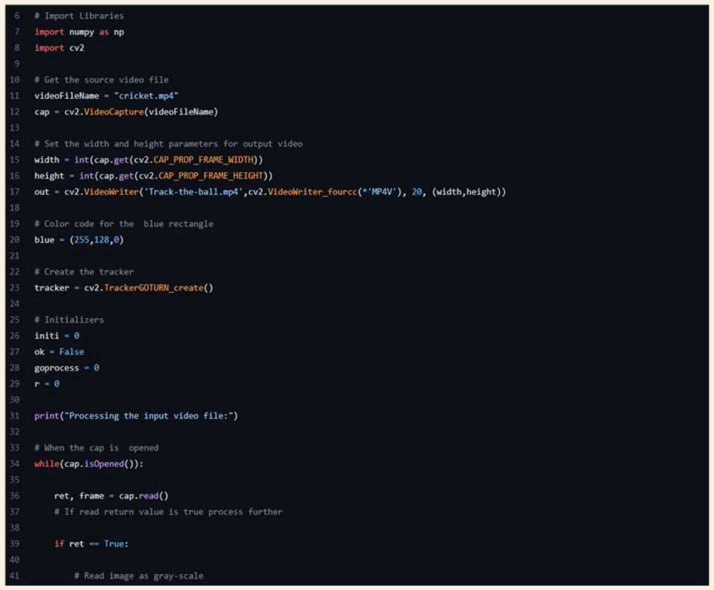
BALL TRACKING



OUTPUT

Ball Tracking

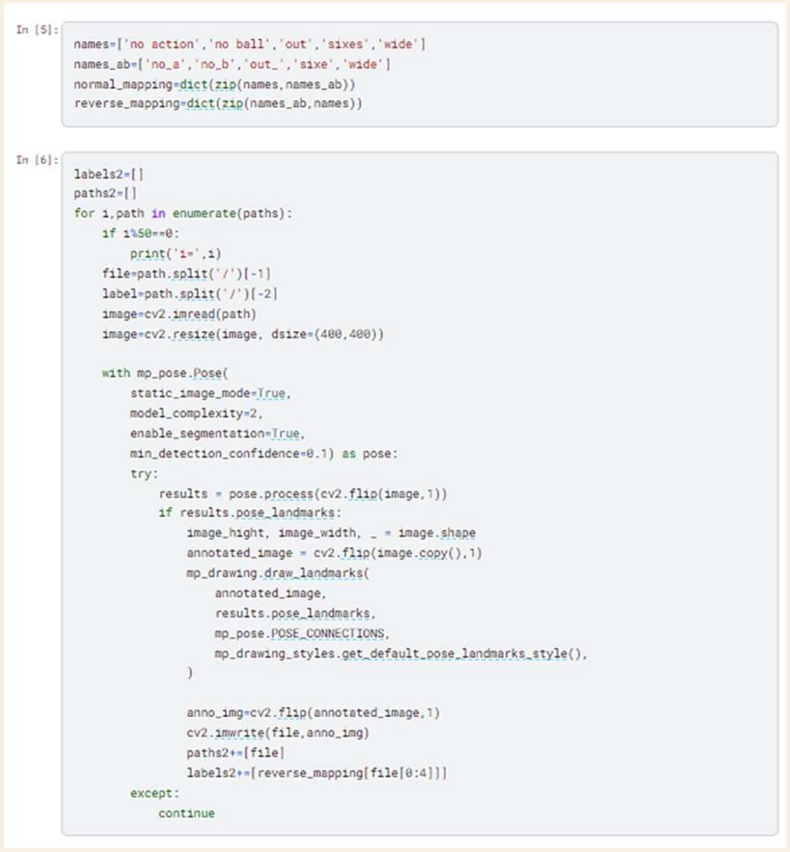
Code Implementation



```python
# Import Libraries
import numpy as np
import cv2

# Get the source video file
videoFileName = "cricket.mp4"
cap = cv2.VideoCapture(videoFileName)

# Set the width and height parameters for output video
width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
out = cv2.VideoWriter('Track-the-ball.mp4',cv2.VideoWriter_fourcc(*'MP4V'), 20, (width,height))

# Color code for the  blue rectangle
blue = (255,128,0)

# Create the tracker
tracker = cv2.TrackerGOTURN_create()

# Initializers
initi = 0
ok = False
goprocess = 0
r = 0

print("Processing the input video file:")

# When the cap is  opened
while(cap.isOpened()):

    ret, frame = cap.read()
    # If read return value is true process further

    if ret == True:

        # Read image as gray-scale
```

UMPIRE DECISION

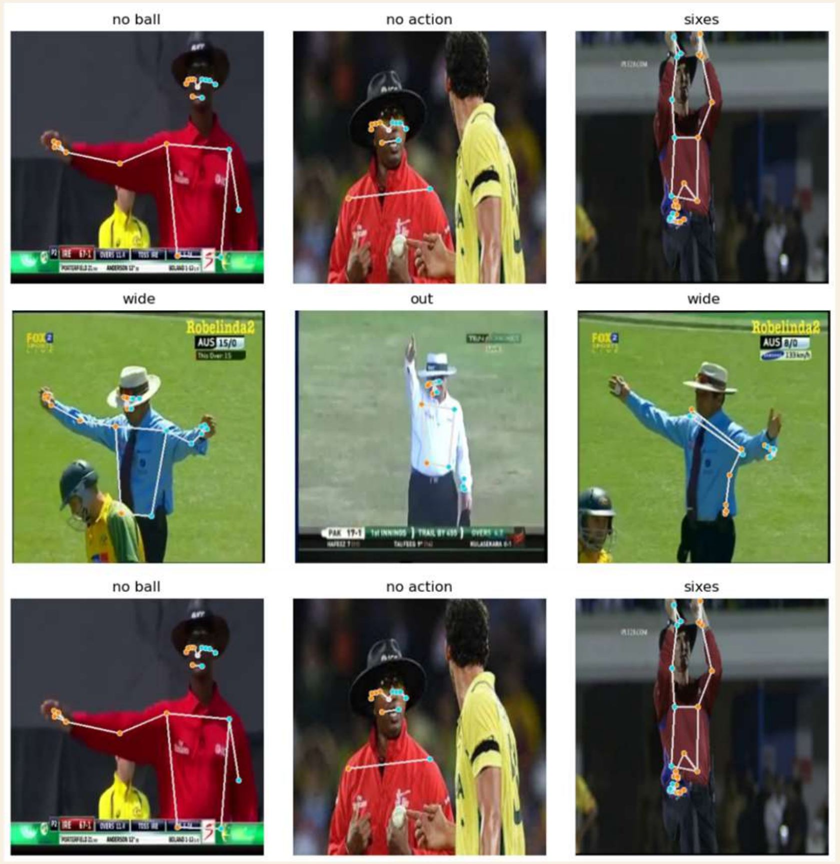
Code Implementation



```
In [5]:
names=['no action','no ball','out','sixes','wide']
names_ab=['no_a','no_b','out_','sixe','wide']
normal_mapping=dict(zip(names,names_ab))
reverse_mapping=dict(zip(names_ab,names))
```

```
In [6]:
labels2=[]
paths2=[]
for i,path in enumerate(paths):
    if i%50==0:
        print('i=',i)
    file=path.split('/')[-1]
    label=path.split('/')[-2]
    image=cv2.imread(path)
    image=cv2.resize(image, dsize=(400,400))

    with mp_pose.Pose(
        static_image_mode=True,
        model_complexity=2,
        enable_segmentation=True,
        min_detection_confidence=0.1) as pose:
        try:
            results = pose.process(cv2.flip(image,1))
            if results.pose_landmarks:
                image_hight, image_width, _ = image.shape
                annotated_image = cv2.flip(image.copy(),1)
                mp_drawing.draw_landmarks(
                    annotated_image,
                    results.pose_landmarks,
                    mp_pose.POSE_CONNECTIONS,
                    mp_drawing_styles.get_default_pose_landmarks_style(),
                )

                anno_img=cv2.flip(annotated_image,1)
                cv2.imwrite(file,anno_img)
                paths2+=[file]
                labels2+=[reverse_mapping[file[0:4]]]
        except:
            continue
```

OUTPUT



PROJECT DEMO

Input Video

- **1Hr 46 mins video**
- 30 **frames** per Seconds
- Match of **India** Vs **SA**
- 20 **Overs** match

Click to Download Video

Drive Link

PROJECT DEMO

Input Video

- **1Hr 46 mins video**
- 30 frames per Seconds
- Match of **India** Vs **SA**
- 20 Overs match

Click to Download Video

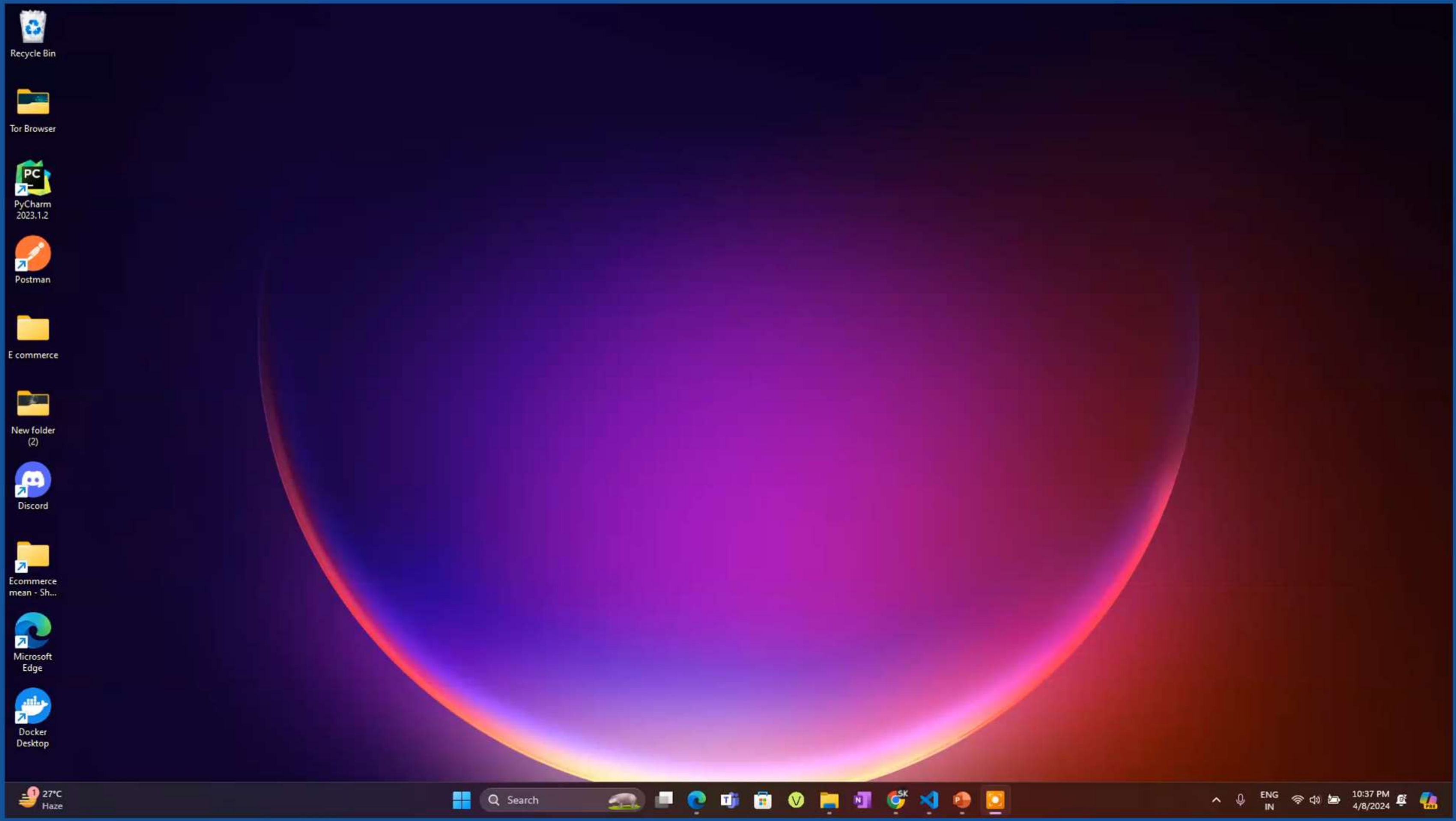
Drive Link

processing...

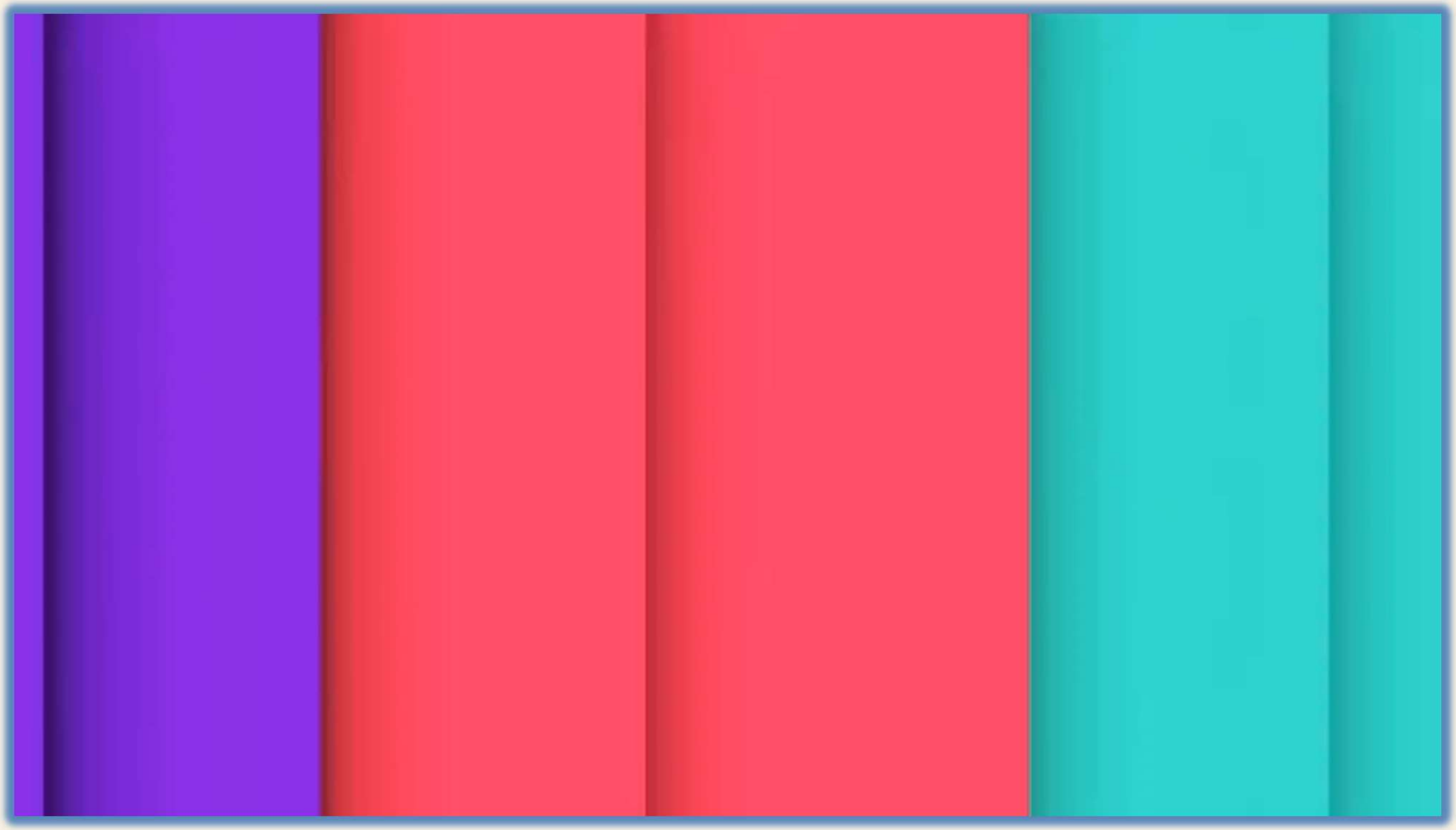
Events are cropped from matches (Saved in Subfolder)

processing...

PROJECT DEMO



FINAL GENERATED VIDEO



PROJECT RESEARCH PAPER

Click to access the research paper

Drive link

THANK YOU