# Detection and Skeletonization of human posture in video

**By:**
Ahmad Nadeem Saigol

# Contents

# Detection and Skeletonization of Human Pose

The goal of the project is to detect human in the video. Once identified, the pose of the human needs to be determined. This problem can be solved using number of different approaches but the one I used, is described below:

## Working of the Code:

The code has a single function which performs the detection and skeletonization of the humans in the video It takes two arguments:

- 'video': This defines the path to the input video, which will act as input.
- 'resize': This takes a tuple of ints (width, height). The input frames are resized to these values. Further, the size of the output frames is also defined by these values.

The video that was used for testing had frames of dimension of (640 x 360), format of '.mov' and frame rate of 25fps. Further these frames, after processing were saved in '.avi' format at 25fps.

- The code makes use of Histogram of Oriented Gradients, which is a feature descriptor. Furthermore, it makes use of Machine Learning technique i.e Linear SVM to detect the humans in a frame.
- The code reads each frame one by one in uint8 format.
- Each frame is resized to specified shape
- As described before, using HOG it detects regions in the frames consisting of humans. It uses a sliding windows size of (4 x4), zero padding of 4 in both directions and another parameter which is set to true which basically groups different close bounding boxes in the frame.
- For each region obtained in the previous step, it is plotted on the original figure and a crop of that region is also obtained.
- This cropped region is converted to grayscale image using the formula:

$$Y = 0.299 * R + 0.587 * G + 0.144 * B$$

- Corners are detected in the cropped region using 'goodFeaturesToTrack()'. This is basically an improved version of Harris Corner Detector as it uses minimum of the eigenvalues (i.e thresholding is applied)
- For each corner obtained in the last step, they are plotted in the original image after adjusting the values accordingly. In the similar fashion, a line is also drawn from the center of the region to these corners. In this way, the pose of the human is determined and plotted on the image.
- It may be noted that outliers were also detected by the algorithm. These were removed by thresholding i.e only those corners were considered whose x (and y) values lie between 15% and 85% of height (and width).

## Few Important Points Regarding Code:

- The code was written in Google Colab in python environment using OpenCV. (.ipynb is provided).

- It was noted that the algorithm is limited to full size humans. That is, it does not work for person whose complete body is not visible in the frame (which can be seen in the last part of the video)
- The code creates a video file which can be used for viewing the results.

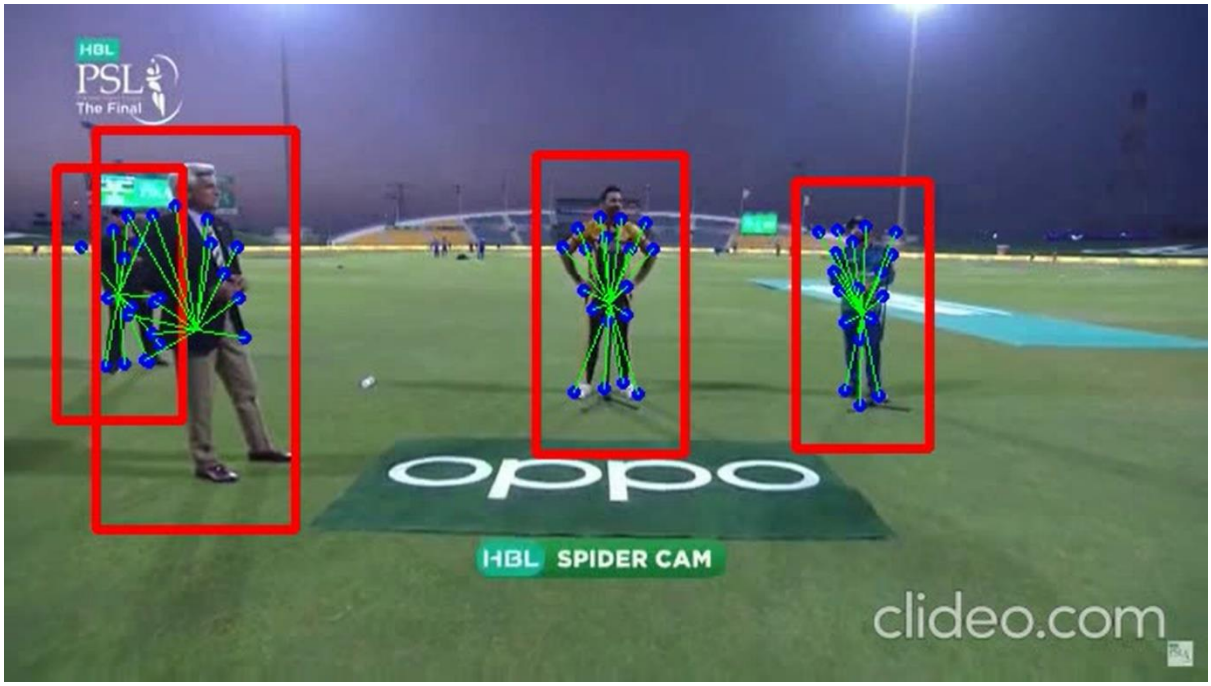
## Results:

### Multiple Persons:

Snippets at time 5s is shown below:

#### Input:


#### Output:




## Single Person:

Snippets at time 8s is shown below:

### Input:


### Output:


## Code:

```
import cv2
from google.colab.patches import cv2_imshow
import sys
import numpy as np

def Detector (video, resize):
  '''
  Detects and skeletonize the humans.
  Also returns the modified video to 'output.avi'

  Parameters:
    'video': str
      path to the video
    'resize': tuple of integers (Width, Height)
      resizes each frame to specified values.

  '''

  # Initializing the HOG person detector
  hog = cv2.HOGDescriptor()
  hog.setSVMDetector(cv2.HOGDescriptor_getDefaultPeopleDetector())


  #for reading frames in video
  cap = cv2.VideoCapture(video)

  if (cap.isOpened()== False):
    sys.exit("Error opening video stream or file")
```

```
  #for storing video
  out = cv2.VideoWriter('output.avi', cv2.VideoWriter_fourcc('M','J','P
','G'), 25, resize)

  while(True):

    ret, frame = cap.read()

    if ret == True:

      #Resizing
      frame =cv2.resize(frame, resize)


      # detect humans
      (regions, _) = hog.detectMultiScale(frame, winStride=(4, 4), padd
ing=(4, 4), scale=1.05,  useMeanshiftGrouping = True)


      # Draw regions
      for (x, y, w, h) in regions:
          cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 0, 255), 3)

          crop = frame[y:y+h, x:x+w]

          #grayscale
          crop = cv2.cvtColor(crop, cv2.COLOR_BGR2GRAY)

          # detect corners with the goodFeaturesToTrack function.
          corners = cv2.goodFeaturesToTrack(crop, 27, 0.01, 10)
          corners = np.int0(corners)

          width, height = crop.shape

          #plot each corner and draw line from centre of rectangle to i
t
          for i in corners:
            a, b = i.ravel()

            #Removing outliers in the crops (rectangles)
            if a < (0.15* height) or a > (0.85*height):
              continue

            elif  b < (0.15* width) or b > (0.85*width):
              continue
            else:
```

```
                #adjusting for original image size
                a = a +x
                b= b + y

                cv2.circle(frame, (a, b), 2, (255,0,0), 2)
                cv2.line( frame, ((np.int(height/2)+x), (np.int(width/2)+
y)), (a,b), (0,255,0), 1)

       #Write the frame into the file 'output.avi'
       out.write(frame)

     else:
       break

  # release the video capture and video write objects
  cap.release()
  out.release()

Detector('/content/Input.mov', (640,360))
```