Automated Exercise Trainer and Game Statistics Application

(Health & Wellness Category)

By:
Parshv Patel

CONTENTS

Chapter 1

Introduction

1.1 Motivation

Today, not considering the quality of coaching that's also a huge issue, most athletes don't get proper coaching due to not enough personal attention (average athletes per coach ratio is 15:1 in developed countries), causing increased injuries, lack of performance, and increased dropouts, unless the athletes are willing to pay exorbitant prices for personal coaching, which is very rare. These problems not only have long term consequences on athletes, but also on their families (20% of american families spend at least $1,000 each month on youth sports). I have a solution to this problem where I will provide a fully automated computer vision and AI-based personal training assistant that tracks and analyzes movements and objects to provide meaningful insights on technique correction, injury prevention, and external performance metrics (shot angle, speed tracking etc.), in real-time. I will improve movement-based tracking by removing sensors and working through minimal cameras. This technology aims to empower the coaches and athletes by giving them detailed insights to train more efficiently.

1.2 Problem Statement

In the current time when exercise and sports are much needed to keep oneself healthy, fit and fine I will start to make an automated exercise trainer application which works seamlessly on a mobile device. An application which is easy to install and which provides insights in real time using mobile camera and the mobile µprocessor. I will also provide various game statistics like the speed, angle, trajectory, bounce point and other metrics involved in the game to the user through a mobile application. The main advantage in using this application is that this application provides all these insights without using any sensors and just using the moderate resolution cameras of the mobile phones mounted at strategic angles.

1.3 Objective

My main objective is to develop an application which automatically detects the pose of a person and gives classification and correction of the detected exercise. I will also provide various game statistics after analyzing the player's posture using Openpose and through various object detection and tracking state-of-art models available in the literature. The various metrics obtained through these models will be analysed using different data science techniques to obtain parameters like speed, angle, trajectory of the object, point of hit, point of bounce etc. I will be using various machine learning and deep learning techniques and algorithms to implement this.

1.4 Scope of Work

The end product of my project is delivering a fitness application wherein the application can detect all exercises using my pretrained model. The only thing the person needs to have is a good camera and my application. The application is scalable in any field of work. It works on the single person pose estimation i.e. the detection of only one person in the frame or in the exercise. The application providing game statistics has some pre-stated limitations that the whole court must be visible in the frame. The players and the object must be in the frame while the shot is being played for the model to detect the location of the object.

1.5 Tools and Technologies

For an automated exercise trainer application I will use OpenPose Library for pose estimation. Also I will use a Dynamic Time Warping algorithm for calculating the distance between testing video and training videos for Human Pose Estimation. I will make the use of Chaquopy Library for making the python code of the project automated exercise trainer run into Android Studio.

For the project Game Statistics I will use YOLO Model for object detection and object tracking. Also I will use OpenCV Library for real time object tracking using the pretrained object models available in the library. The use of Depth Estimation Algorithm will be made for the calculation of speed, point of hit and bounce point. For graphically representing the bounce point in the court I will use perspective transform technique called homography transform which converts 3D court into a 2D representation. Player detection will be done using detectron which is a pre-trained library by FacebookAI.

Chapter 2

Literature Survey

This literature survey shows the implementation of two different projects namely Automated Exercise Trainer and Game Statistics. Both of these projects are developed using Machine Learning and Deep Learning techniques. Both of these projects are included in the application.

The Automated Exercise Trainer project includes the pose estimation, exercise detection, classification and correction of exercises, extraction of body angles, scoring of exercises based on performance and calculation of various other parameters such as Rep Count. This project makes the use of various libraries and machine learning algorithms. The model will be trained for a long time to get the desirable output. The Openpose library will be used for the human pose estimation. This is a single person human pose estimation model using openpose library. Also I will use a Dynamic Time Warping algorithm for calculation of the cost between testing video and training videos. This algorithm is used for 2D Pose Estimation as well as 3D Pose Estimation.

The Game Statistics project includes the ground mapping or court mapping and calculation of some parameters used in games such as point of hit, point of bounce, speed, angle, trajectory etc. Here I will identify and calculate several game parameters. Also the project Game Statistics includes object tracking and object detection. This project also includes use of several libraries such as OpenCV for object tracking. The YOLO Model will be used for object detection. And also several machine learning algorithms will be used such as Depth Estimation Algorithm for the estimation of several game parameters.

Both these projects make very good use of Data Science concepts. Data Gathering and Data Preprocessing is also a part of both these projects. The dataset used in both these projects will be prepared by me. I will gather videos of many exercises such as pushups, plank, wallchair etc as a dataset for our project Automated Exercise Trainer for testing the model. Also the videos of games such as badminton and tennis will be used for testing the Game Statistics model.

Hence, both these projects will be made creating my own dataset i.e. using my own training and testing videos. Both the projects i.e. Automated Exercise Trainer and Game Statistics will be implemented in an Android Application. This application will be used by any person to perform the exercise remotely, just the person needs to have a good camera and our application.

Chapter 3

Preliminaries

3.1 YOLO Model

YOLO is an acronym for 'You Only Look Once'. YOLO is used to detect objects using its pretrained model and it puts bounding boxes on the detected objects. It is based on a deep convolutional neural network also known as deep CNN. YOLO is a fully convolutional network as it has only convolutional layers, 75 Conv. layers to be precise. It has no pooling layer. It uses skip connections also known as residual connections in the network. It uses a unique technique for downsampling i.e. it uses conv. Layer with stride 2 which is beneficial in avoiding loss of low level features in the feature map. The YOLOv3 architecture is as shown below.

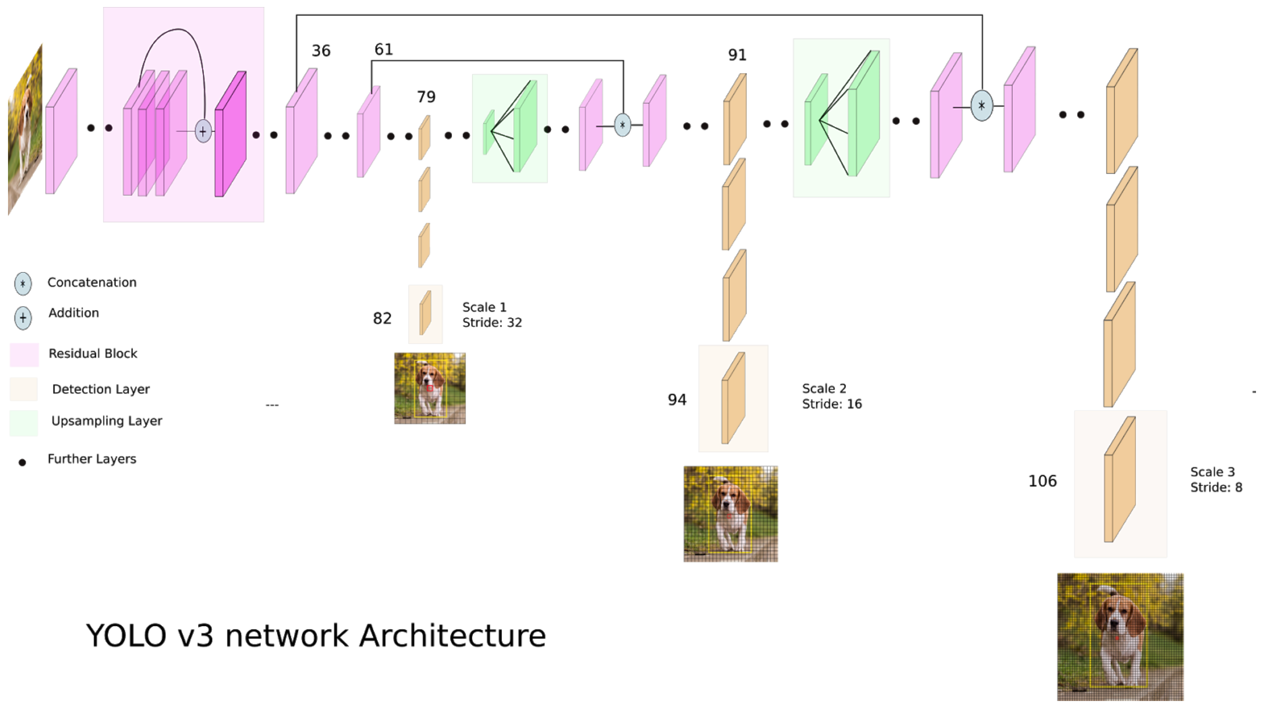


Fig 3.1.1 YOLOv3 Architecture

The working of YOLO is quite interesting as it only looks once in the image as the name suggests. The first task is to divide the image into grids of 13x13. Then each cell is assigned a bounding box that has its own confidence score and the class it predicts. This generates a very large number of bounding boxes as shown in the figure.

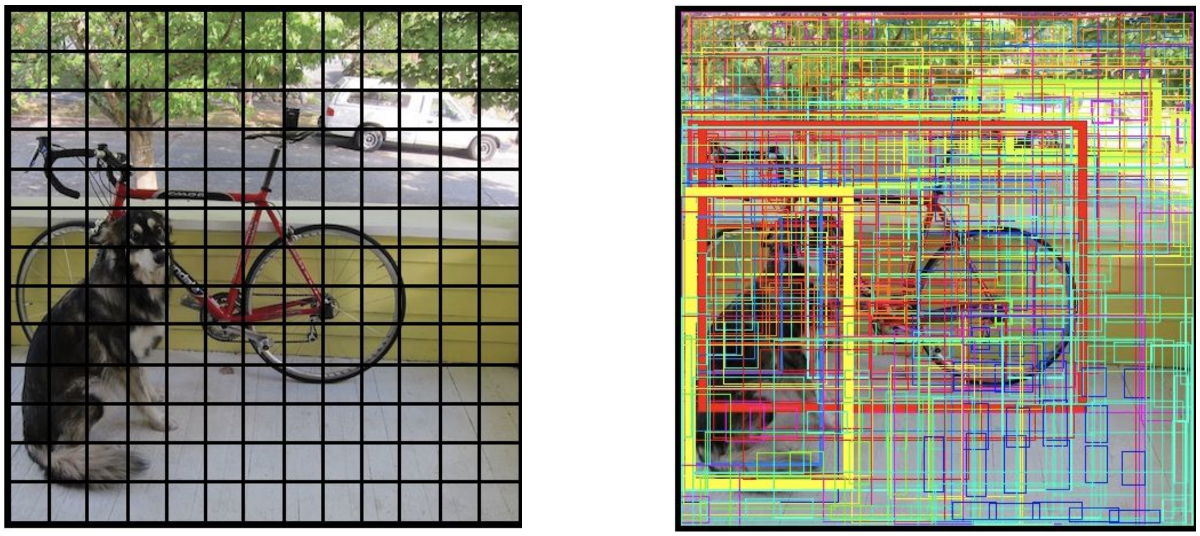


Fig 3.1.2 Object Detection using YOLO

The predicted cell represents a class which in our case is the object used in playing in the game. There are 13x13= 169 cells which have confidence values. Then out of the probability values a threshold value is selected and the bounding boxes having values less than the threshold value are ignored. Then based on the IOU value the best suited bounding box is chosen and is displayed as the output with the probability of the class and the bounding box. The output of the given image is as shown in the figure.

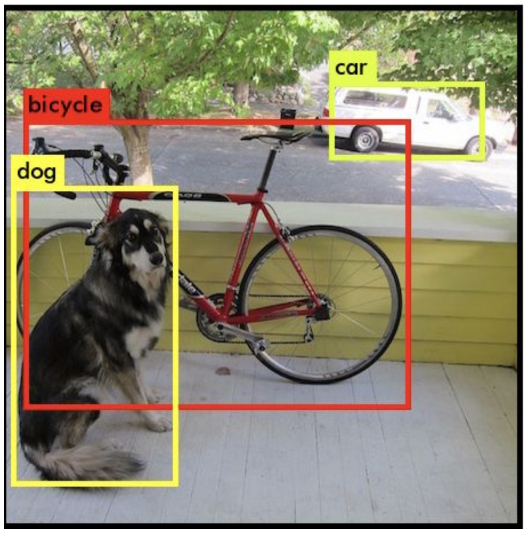


Fig 3.1.3 Bounding Box

3.2 Openpose

Openpose is a python library used for human pose estimation. This library represents the first real time multi-person system for human pose detection. Openpose detects the human body, hand, facial and foot keypoints on images. It detects the human body in total 135 keypoints. The functionality of Openpose includes both 2D real-time multi-person pose estimation and 3D real-time single-person pose estimation. The input here can be image, video, webcam, Flir/Point Grey and Ip camera. And the output can be a basic image plus keypoint display/saving or keypoints in the form of array class. In our project Automated Exercise Trainer I will clone the Openpose Github Repository. Also we can unzip the Openpose zip folder by mounting the drive in google colab.

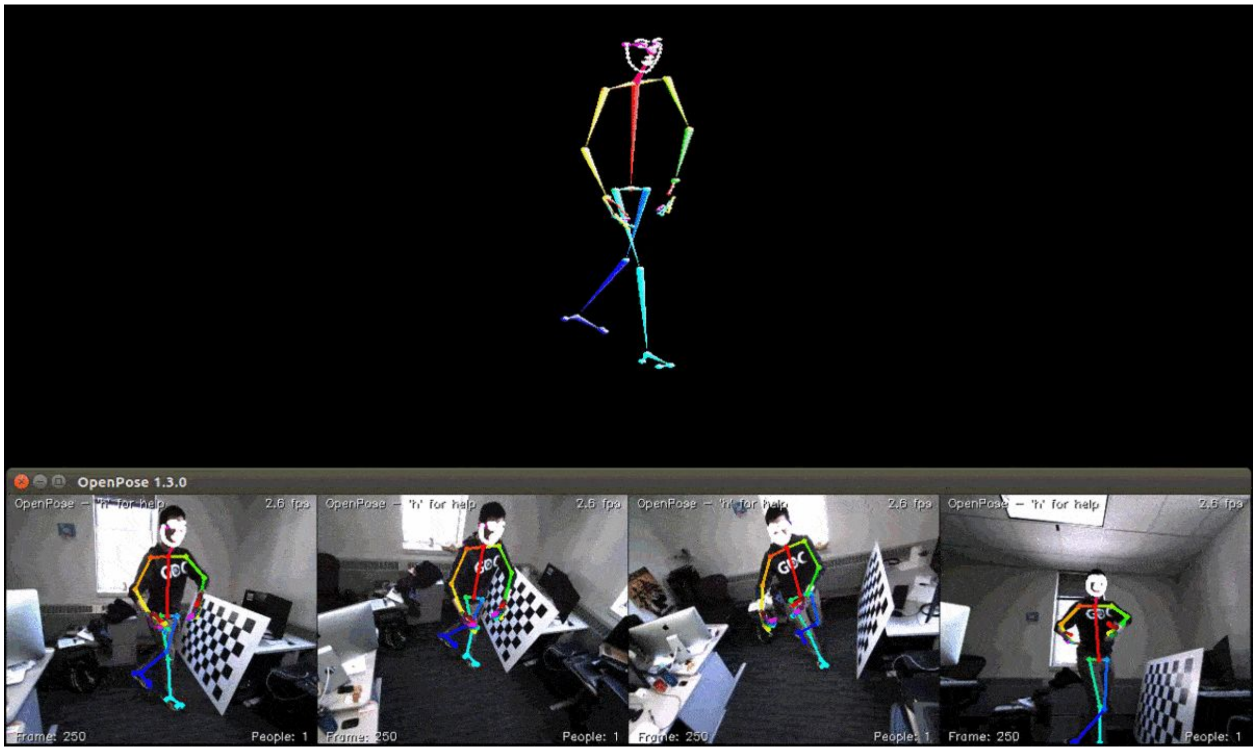


Fig 3.2.1 Pose Estimation using Openpose

3.3 Dynamic Time Warping Algorithm

Dynamic Time Warping algorithm is used to measure the similarity or measure the distance between two arrays or time series of different length. In time arrangement examination, Dynamic Time Warping (DTW) is one of the calculations for estimating similitude between two transient groupings, which may differ in speed. DTW has been applied to fleeting arrangements of video, sound, and design information — surely, any information that can be transformed into a straight grouping can be broken down with DTW.

In our project Automated Exercise Trainer I will be using a Dynamic Time Warping algorithm for comparing the test video with the training set of clips. After that I will be performing normalization by eliminating time as a constraint. And then afterwards I will be performing comparison and assignment of scores using Dynamic Time Warping algorithm. Hence, Dynamic Type Warping algorithm is used to calculate the cost or distance between testing video and training videos.

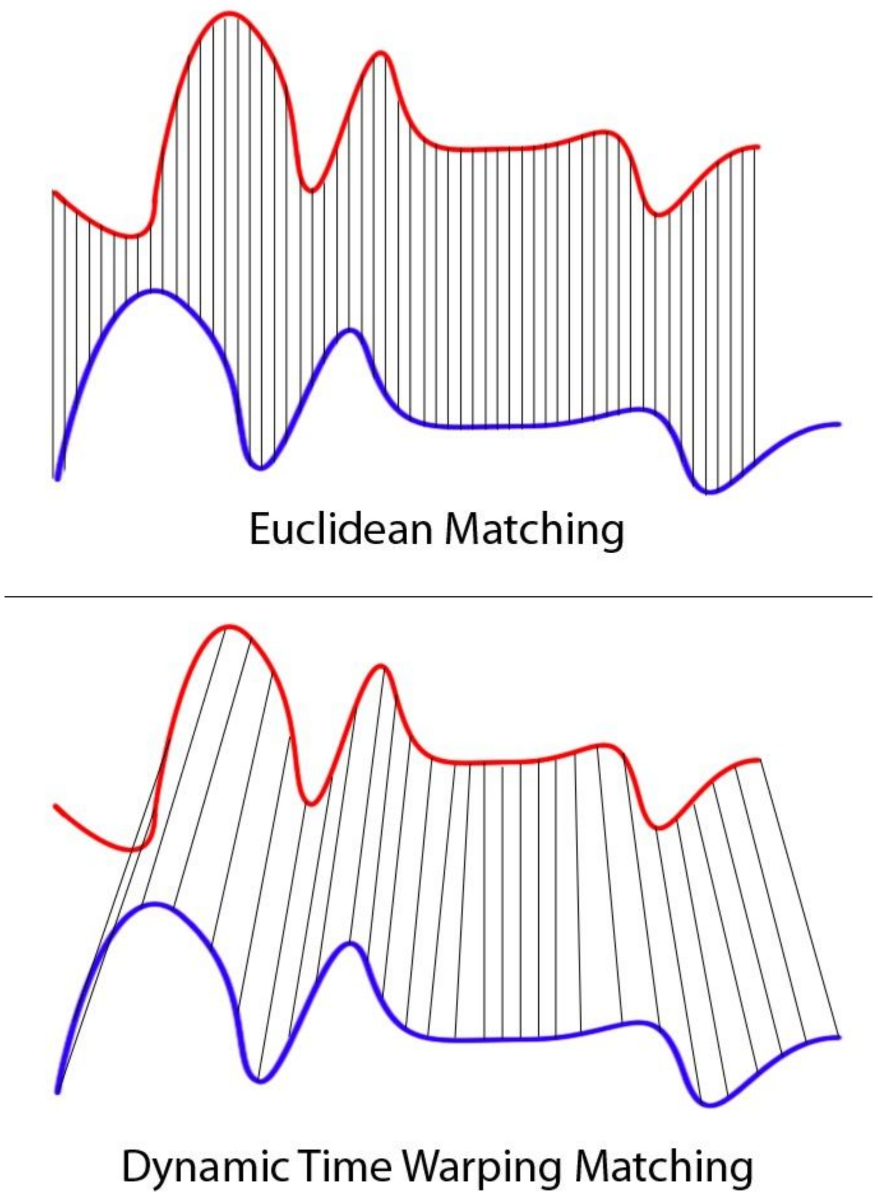


Fig 3.3.1 DTW Matching

3.4 Homography

Homography transform is a perspective transform technique which uses a mathematical formula to transform the points from one image to another image. It converts a planar surface from an image to a different plane with the same relative dimensions. It requires 4 or more points to identify a plane from the image. If there are more than 4 points it is better to transform the image as it gives more elaborate dimensions of the plane. Homography transform is available as a function in OpenCV. It tries to best fit the input points passed into the function, based on the mathematical formula as shown below.

$$H = \begin{bmatrix} h_{11} & h_{21} & h_{31} \\ h_{12} & h_{22} & h_{32} \\ h_{13} & h_{23} & h_{33} \end{bmatrix}$$

$$s\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = H\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Fig 3.4.1 Homography Matrix

These equations are only valid in case the points in the image are on a single plane. Otherwise the output will be a distorted one. One popular application of homography is that it is used in panora to align the images correctly which are further stitched together to form an image.

3.5 Chaquopy Library

In our project Automated Exercise Trainer, Chaquopy library will be used to run python code in Android Studio. I will use this library to make our python code run into Android Studio. Chaquopy allows us to mix python, java and kotlin in any application, using whichever language is best for our needs. I will make the python code of our project Automated Exercise Trainer run in Android Studio so that once the code runs in Android Studio and gives the output in Android Studio console, we can design the SDK for our fitness application. So, I will use Chaquopy library for making the python code run into Android Studio.

3.6 Depth Estimation Algorithm

Depth Estimation is a technique to obtain the spatial representation of an image to recover the unknown insufficient information from a 2D image. There are several techniques used to get the depth estimation value from an image like the classical stereo method and the new Deep learning methods. I will be using the DL method of self-supervised monocular method for depth estimation. In this method first the given image is converted into a single color image with different intensity values also known as depth maps. This network is pre-trained on the state of the art KITTI dataset. Then from the depth map generated the depth is estimated based on the intensity of the colors using mathematical formulas and by minimizing the regression loss. The figure below shows the depth map generated.

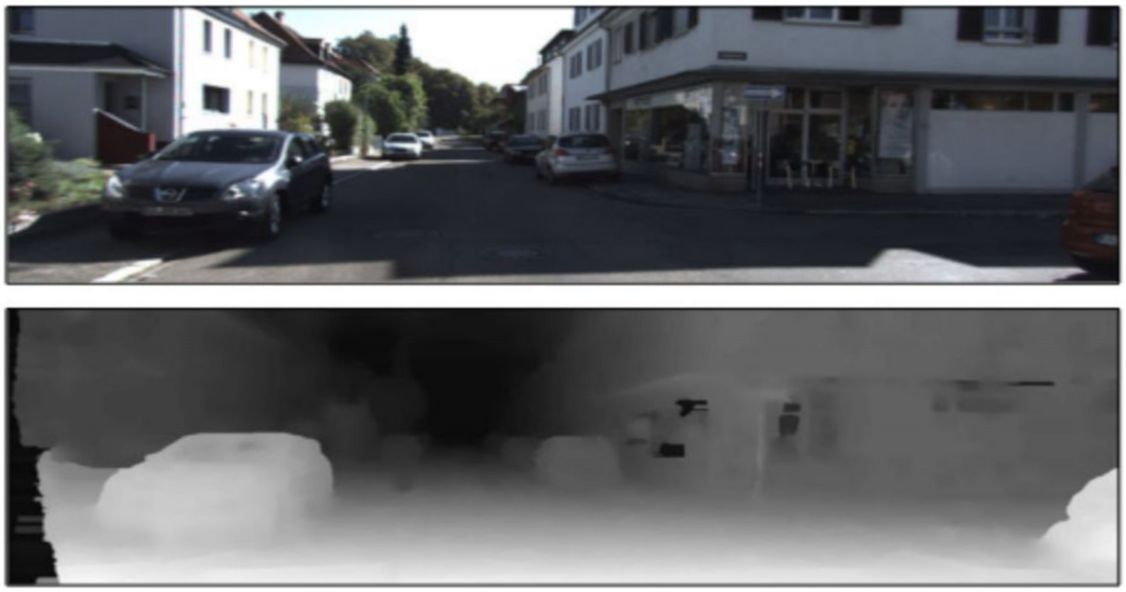


Fig 3.6.1 Depth Estimation Algorithm

Chapter 4

Data and Study Area

4.1 Data Generation / Gathering

In both the project i.e. Automated Exercise Trainer and Game Statistics, I will create my own dataset. I will take the videos of various exercises for the Automated Exercise Trainer project to train the model. Nearly 500 videos of various exercises will be taken by me to create the dataset for the project Automated Exercise Trainer. All these videos will be used by me for the training purpose as well as for testing the model.

I will create npy files out of these videos as I can’t give videos as input to the model. I will convert all the videos of exercises into respective npy files. And those npy files will be given as input to the model for training and testing purposes. So these npy files will be used as the dataset for the project Automated Exercise Trainer. Hence I will create the dataset on my own for the project Automated Exercise Trainer.

Similarly, for the project Game Statistics I will record and gather videos of several games for training the model. Also these videos will be used by me for testing the model. I will take the videos of Badminton and Tennis and these videos will be used by me for training the model. Then the taken videos will be converted into frames and objects will be labeled using the labelIMG application. Then the main goal will be to fine tune the pretrained model available in the YOLOv3 model trained on the COCO dataset. Further these models will be trained over and again with the different types of images to overcome false positives and to reduce the losses in prediction.

4.2 Data Pre-processing

In any Machine Learning process, Data Preprocessing is that progression wherein the information gets changed, or Encoded, to carry it to such an express, that now the machine can without much of a stretch parse it. As it were, the highlights of the information would now be able to be effectively deciphered by the calculation.

Data preprocessing is a data mining procedure that includes changing crude information into a reasonable organization. Genuine information is frequently deficient, conflicting, as well as ailing in specific practices or inclines, and is probably going to contain numerous blunders. Information preprocessing is a demonstrated strategy for settling such issues.

In Real world data are commonly deficient: lacking characteristic qualities, coming up short on specific properties of intrigue, or containing just total data. Loud: containing blunders or anomalies. Conflicting: containing disparities in codes or names.

Data preprocessing/planning/cleaning is the way toward distinguishing and revising (or expelling) degenerate or mistaken records from a dataset, or and alludes to recognizing inaccurate, deficient, unessential pieces of the information and afterward altering, supplanting, or erasing the grimy or coarse information.

After creating my own dataset for the project Automated Exercise Trainer and Game Statistics, I will preprocess the dataset by removing the noise in the dataset. The videos I will take to create my dataset will be trimmed and processed so that the noise in the dataset gets removed.

Chapter 5

Implementation

5.1 Automated Exercise Trainer

5.1.1 Pose Estimation

Pose estimation is basically detection of keypoints on the human body. So the first task of my project Automated Exercise Trainer will be detection of keypoints on the body of the person who performs the exercises i.e Human Pose Estimation. I will be performing pose estimation on the videos I will take to create the dataset.

I will perform 2D real-time multi-person keypoint detection using Openpose Library. Openpose library will be used to perform pose estimation on the videos of several exercises. Total 18 keypoints will be detected on the human body using the Openpose Library. Openpose Library is used for 2D real-time multi-person keypoint detection as well as 3D real-time single-person keypoint detection. We can perform 15 or 18 or 25-keypoints body/foot keypoint estimation using Openpose Library.

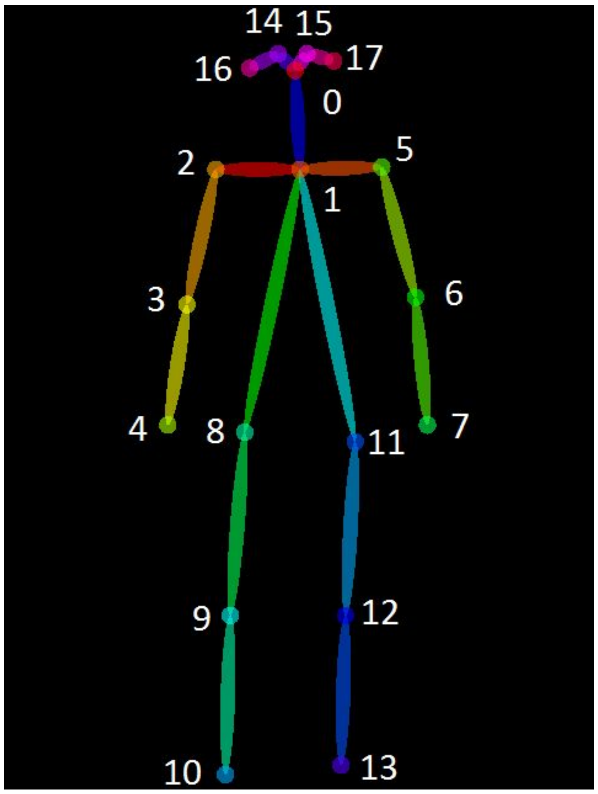


Fig 5.1.1.1 18 Keypoints using Openpose

5.1.2 Exercise Classification

Here, for our project Automated Exercise Trainer after giving videos as the input to the model we will get the exercise being performed in the video as the output. Hence I will classify our dataset of videos into different exercises being performed in the videos. I will give one video of any exercise as input to the model and the output we will get is the exercise detected from that video. Hence different exercises will be classified as per the videos given as the input to the model. First I will determine the pose estimation of the input video using Openpose Library and then after I will determine the exercise being performed in the video.

First of all I will convert all the input videos into respective npy files and those npy files will be used as the dataset to train the model. Then we will use a Dynamic Time Warping (DTW) algorithm to calculate the distance or cost between the test video and training set of clips. Deep Learning Framework Caffe will be used to train the model. It is written in C++ Language, with a python interface.

Also exercise nature will be detected whether the exercise performed is good or bad. Some body angles will also be calculated namely ‘Upper arm and torso angle range’ and ‘Upper arm and forearm minimum angle’. Feedback to the exercise performed will also be given i.e. how the exercise can be improved will also be given as the output. The detection of the exercise arm will also be done, that is the exercise is performed using which arm will also be detected. Hence there are many things that will be detected using video of the exercise as the input to the model.

5.1.3 Exercise Scoring and Correction

After performing the detection of exercise I will do scoring of each exercise and will provide proper feedback to improve the exercise being performed. I will show the good and bad scores of each exercise in the form of an array. I will correct the exercises by providing proper feedback if the exercise is being performed incorrectly or improperly. The scores will be given by calculating DTW distance using Dynamic Time Warping algorithm. And the feedback will be provided based on the scores given to the exercises being performed.

After providing good and bad scores to the exercises, I will scale those scores in the range of 0 to 10. If the exercise is being performed in a correct and proper way then the score of that exercise should be above 5 out of 10 that is between 5 and 10 (including 10). If the exercise is not performed in a correct way or is being performed in an improper way and needs proper feedback to improve then the score of that exercise should be below 5 out of 10 that is between 0 and 5 (including 0). So I will scale the good and bad scores of each exercise out of 10.

5.1.4 Exercise Repetition Count

After scoring each exercise and performing correction of each exercise, I will perform repetition count of each exercise. So basically in push-ups we have a parameter called as Rep Count so that is being calculated if the exercise being performed is push-ups. For example, let's say if a person is performing push-ups exercise and if he/she does 10 push-ups correctly then the value of Rep Count should be equal to 10.

So, let's say if we talk about the exercise push-ups then the rep will be counted only if the shoulder goes below the knee. So this logic of push-ups will be stored in a CSV file. Likewise the logic behind the rep count of every exercise will be stored in a single CSV file. Hence, one CSV file will contain the logic behind the rep count of all the exercises being performed. The CSV file will contain four arguments i.e. the first argument is the name of the exercise, the second argument is the static keypoint (here in the case of push-ups the static keypoint is Knee), the third argument is the dynamic keypoint (here in the case of push-ups the dynamic keypoint is Shoulder) and the fourth argument is the updown (denoting the movement of the body i.e. up or down). Hence, the CSV file will be read in the code to give the output of Repetition Count for any exercise.

5.1.5 Python to Android

So, till now what I did for the project Automated Exercise Trainer will be in Python. Now, to make an Android Application i.e. a Fitness Application, we need to implement the python code into Android Studio. I will use Java as the backend language in Android Studio. So, implementing the python code in Android Studio will be done using Chaquopy library. I will use Chaquopy library to make the python code run into Android Studio.

The library allows us to mix python, java and kotlin in any application, using whichever language is best for our needs. I will make the python code of the project Automated Exercise Trainer run in Android Studio so that once the code runs in Android Studio and gives the output in Android Studio console, I can design the SDK for this fitness application. All the npy files created out of videos of exercises will be put in the Assets folder in the Android Studio so that the code can read the respective npy file out of the Assets Folder. So, I will use Chaquopy library for making the python code run into Android Studio.

5.2 Game Statistics

5.2.1 Court/Ground Mapping

The idea is to develop a solution for mapping the court obtained through the image using a perspective transform like homography transform so that it can be used to represent precisely the various points on the court to show where the ball bounced from the ground and other such metrix. Further the court/ground mapping will be extended so that it can be used to show the player positions based on the mask-RCNN segmentation in sports like basketball and football. Thus this helps in 2D mapping of the points from the video.

Thus the first step is to obtain a frame from the video which is used to obtain edges from the images using a hough line transform by masking with appropriate colors. Here the image is first applied to some OpenCV function and then it is passed through an edge detector with a given threshold of some fixed values.

This detects the edges which make up the court. Then comes the important part of detecting the corners of the court which is detected using some function of CV2 library. Finally the input image is warped on a destination image placing the corners of the image obtained through homography so that it can be used to show the output of the points in a graphically more accurate way.

5.2.2 Object Detection and Tracking

In any sport there are certain objects like the racket, tennis ball, basketball, football etc which are of key importance while providing insights about the game. In my project I will be focused towards badminton and tennis. Thus the objects at hand will be tennis balls and shuttle respectively. Pre-trained models are available for tennis balls but the same are not the case for shuttle as not much research has taken place in that direction nor there are products available in the market focused towards badminton. Thus I will have to fine tune some pre-trained models and find out the best fit for my project. The first task will be to get the dataset. For this I will take some videos on my own, download some match footage and other few pictures from the internet around 1200. Then I will apply some data augmentation techniques on the dataset. I will divide the dataset into training and validation sets. Then I will label the images using the labelImg tool available in open domain.

The then labeled dataset is given to the YOLO pre-trained model on COCO dataset for fine tuning for 15 epochs. The YOLO model will be loaded using the ImageAI library which provides an easy method to load and train various models like YOLO, Inception, RetinaNet and a few more.

Thus a weights file will be generated which is to be given to a code customised for object detection which will detect the object and further layers can be added to the code to get the desired output from the detected model. The detected model gives the set of coordinates of the detected object which can be further used to find the point of bounce and point of hit with the racket. These coordinates are stored in a csv file which can be used further for finding speed, angle and other parameters.

5.2.3 Speed and Angle Calculation

The object detection gives the output in the form of coordinates which are stored for further use. These coordinates are then passed to the depth estimation model which gives the estimated depth of the object in each frame of the video. The depth estimation is done after changing the code available in the public domain according to our needs.

This returns the value of the depth. The value of estimated depth, set of coordinates and some other parameters are used for further calculation of speed and angle. The code and other sensitive information cannot be further disclosed. The distance and speed are calculated using certain formulas of mathematics using the depth estimation value, the coordinates, frames per second value and certain camera calibration values obtained through real-time measurements.

The measurement of the speed is done based on the newtonian formula of speed is equal to distance upon time. Here some mathematical formulas and algorithms are used to obtain the distance and respective time for the calculations.

5.2.4 Point of Hit and Bounce Point

I will train the model for object detection and tracking. The coordinates obtained from the object detection model are used to obtain the trajectory of the object. Now looking at the placement of the camera in the court, we can see that the camera is placed behind the player. Thus the trajectory of the ball will be completely visible and saved. But as the video is recorded in 2D it is a hard task to obtain the point where the ball hits the racket. For this some logic had to be thought and implemented which does not add much overhead to the system and gives fairly accurate results.

The ball is tracked and the list of coordinates is passed to the depth estimation model. The depth estimation model finds the depth of the contour thus formed based on the list of coordinates. The depth estimation function returns the depth of the ball in each frame of the image which is stored in a csv file. These values of depth estimation are then passed to a code which is developed for finding out the specific frame number where the ball/shuttle hits the racket. This returns the coordinates of the point of hit and the frame number in which the ball hits the racket.

Point of bounce cannot be found based on the approach mentioned for finding the point of hit. For this a different approach has to be designed. First the video has to be segmented such that it has two consecutive shots from the players, which ideally has one point of bounce. Then the video is taken and the coordinates of the object are found and stored. These coordinates are then passed through a logic designed to obtain the point of bounce, it coordinates in the image and the number of the frame where the ball bounces on the ground. Further these coordinates are passed so that they can be merged with the court obtained through homography.

Chapter 6

Results and Validations

6.1 Perspective Transform

The Homography transform is a perspective transform technique which converts the given image of court into 2D court. First step in this process is to obtain the edges from the given image.

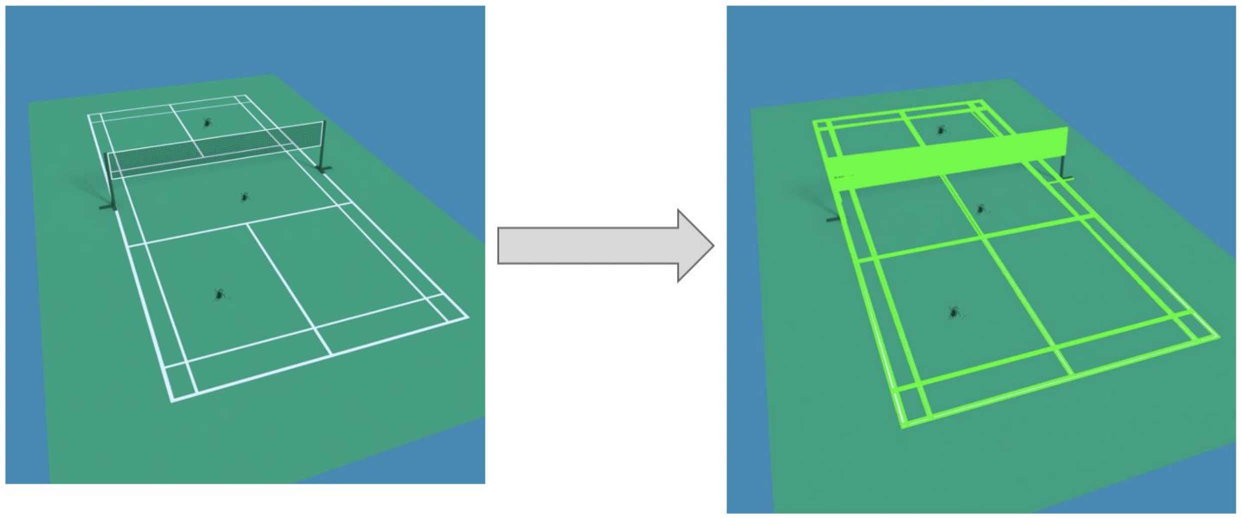


Fig 6.1.1 Homography Transform

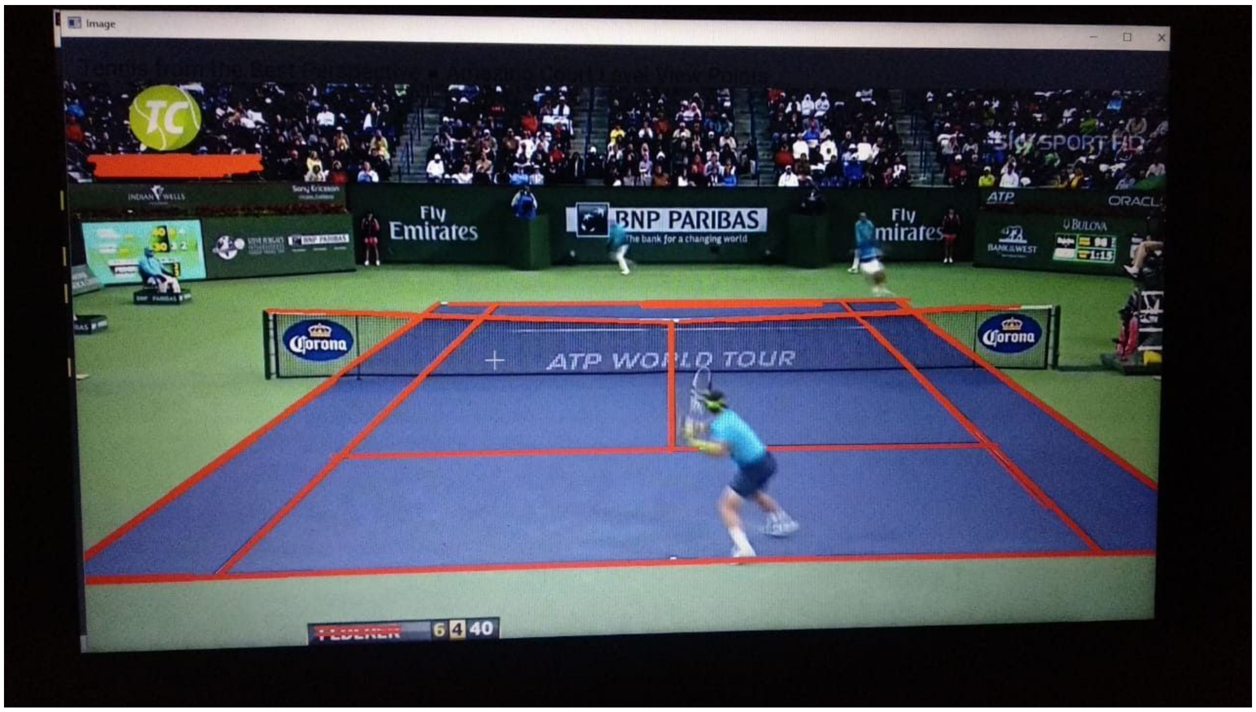


Fig 6.1.2 Perspective Transform

The output when tested on a real time image is as shown after applying the edge detector after certain modifications with the image.

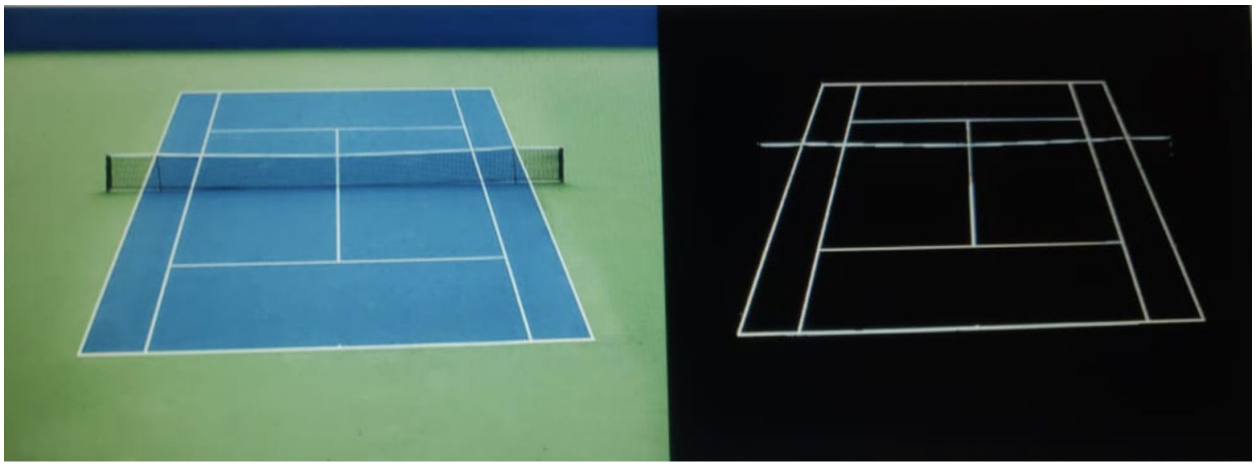


Fig 6.1.3 Court Mapping

The input image and the output obtained after the following transformation is as follows.

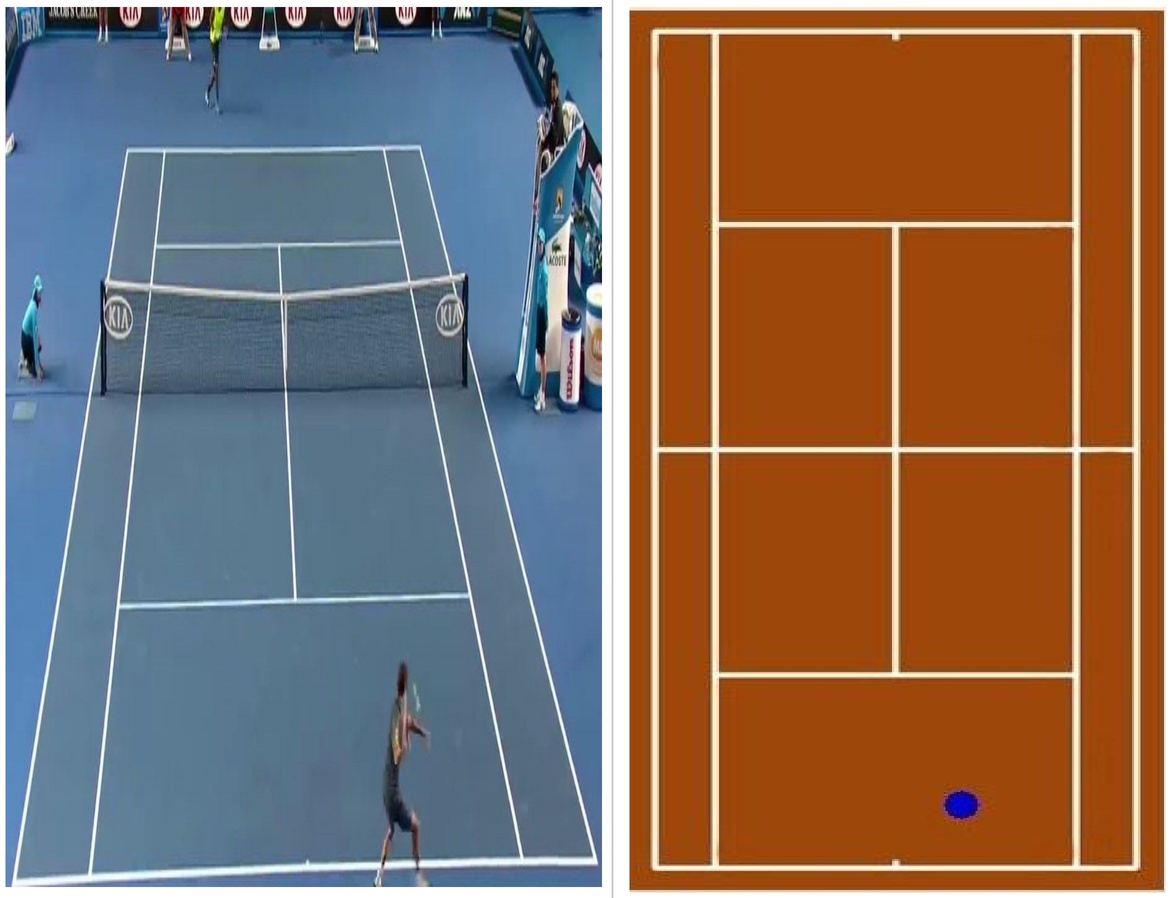


Fig 6.1.4 Point of Bounce

The left image shows the real match picture where the ball bounces on the ground and the right image shows the point on the animated court obtained after court mapping and object detection and after combining their output based on the program developed.

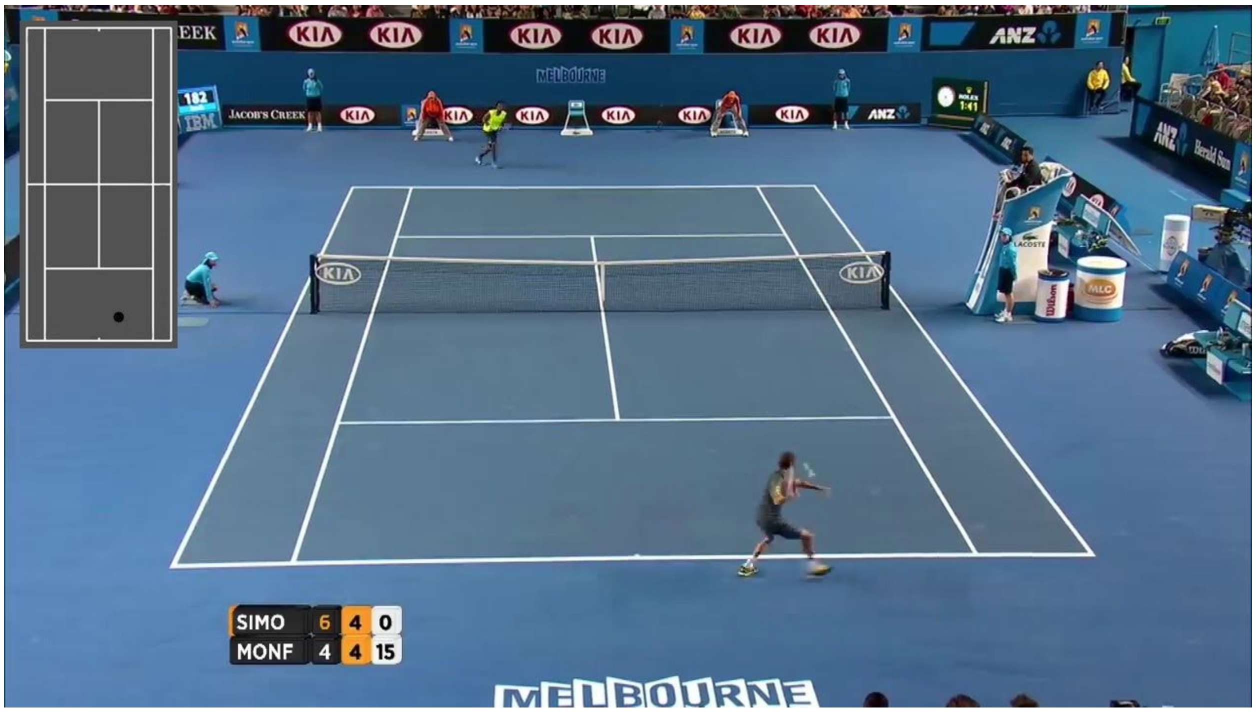


Fig 6.1.5 Final image

Final output after completion of the whole process will be as shown in the figure. The top left corner part contains a graphical image with the point where the ball bounces after hitting the ground.

6.2 Object Detection and Validation

A video is given as input for object detection. The video is then converted into frames and each frame is then given for object detection and respective coordinates and certain parameters are printed as shown in the image below.

The object detection model trained on the pretrained images gives the following output for a badminton match image. The object detected here is ‘shuttle’. The probability percentage is fairly less so I had to suppress all the other classes in the model. I also increased the number of images through data augmentation.

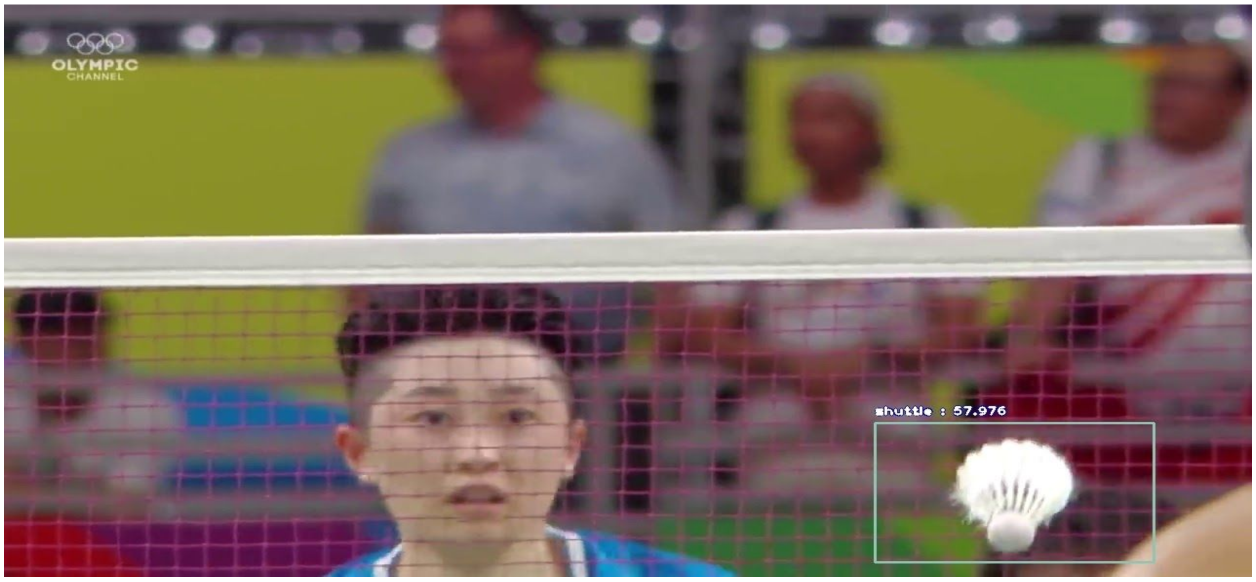


Fig 6.2.1 Object Tracking

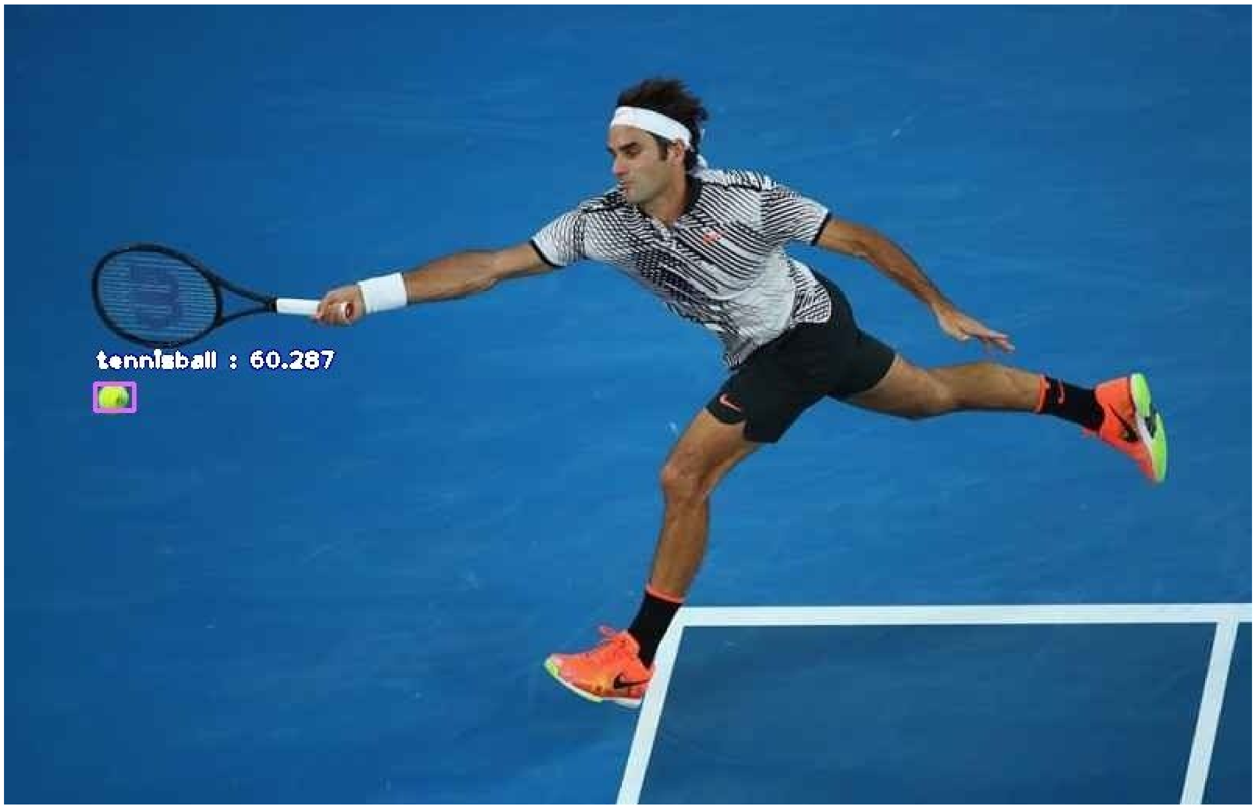


Fig 6.2.2 Object Detection

Again the YOLOv3 model was fine tuned for the tennis ball dataset. The probability is 60% in the first attempt. Further the model was trained and the detection probability was improved.

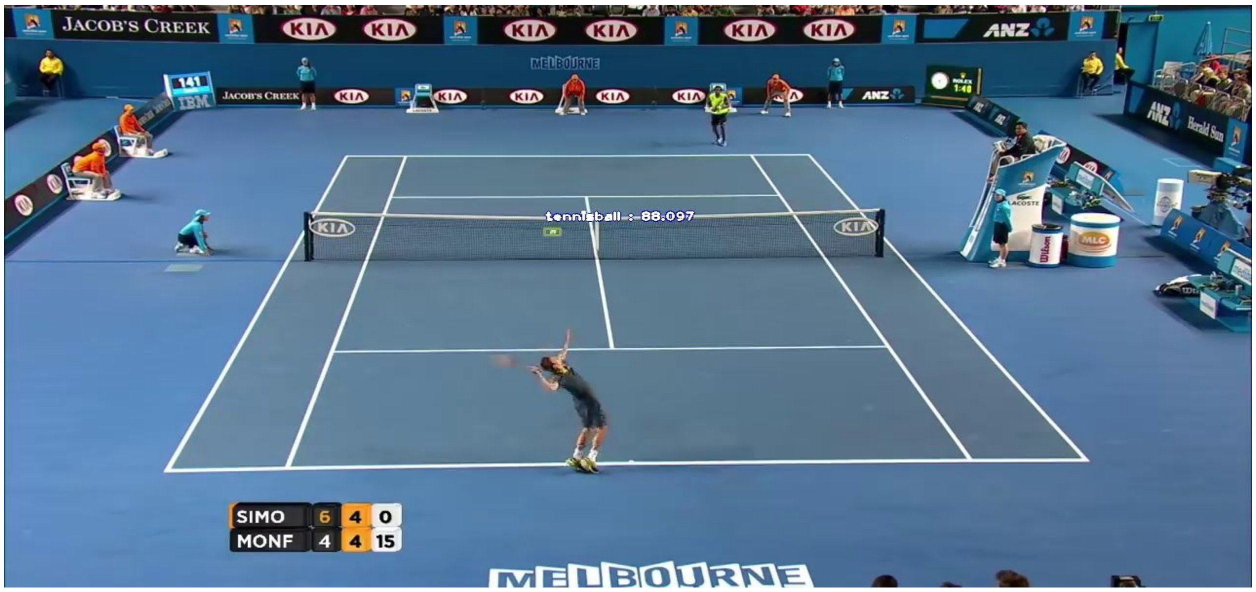


Fig 6.2.3 Object Detection

Thereafter the dataset was increased with some new images. Also data augmentation was done. The output probability of the ball increased. The detection probability is 88%.

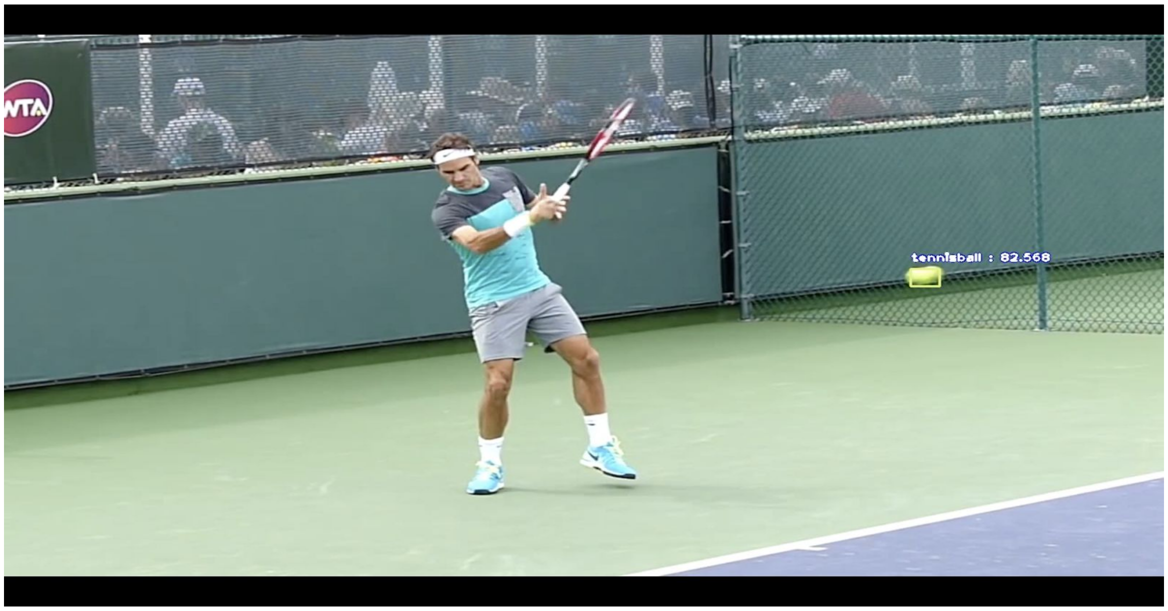


Fig 6.2.4 Object Detection after Data Augmentation

6.3 Speed and Angle

The speed and angle of the ball after passing it through the object detector, depth estimator and the codes designed for calculating speed and angles gives the following values of the object as shown in the bottom left corner. The speed(km/hr) is 7.724. The angle is 66.4°.

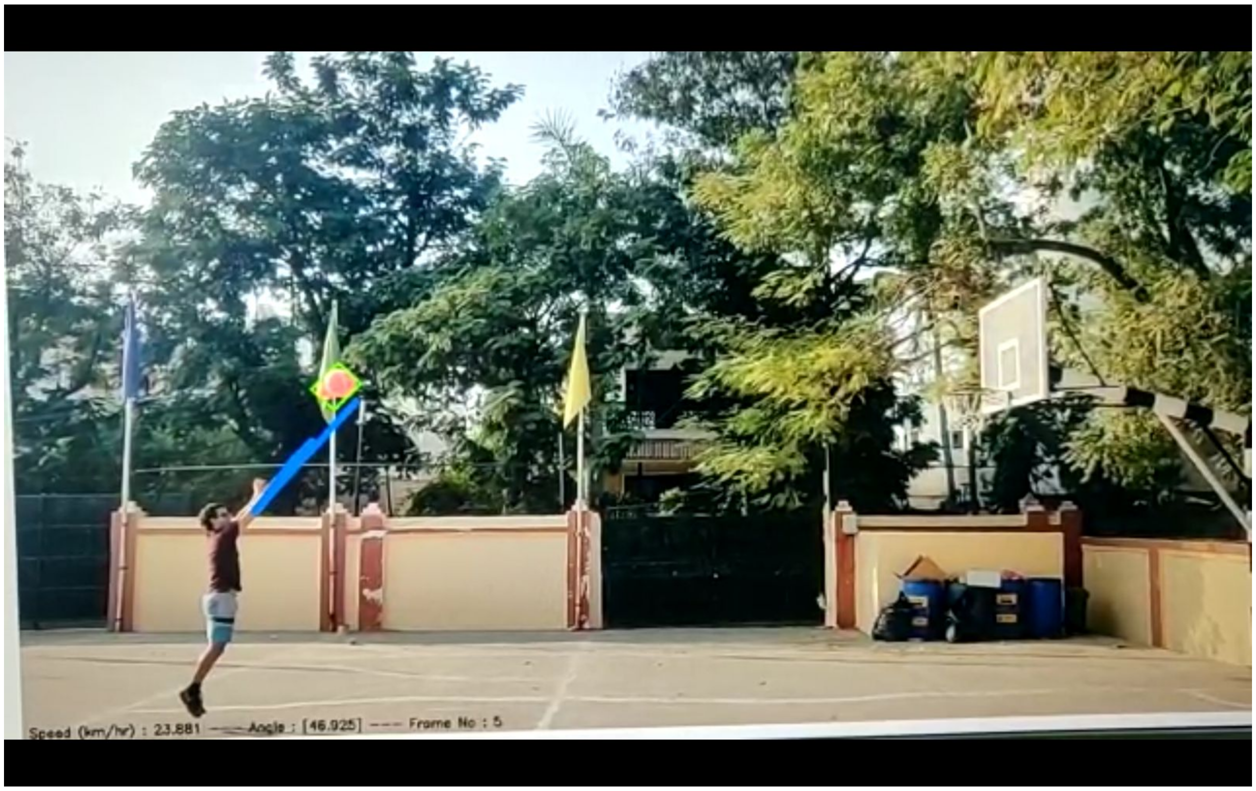
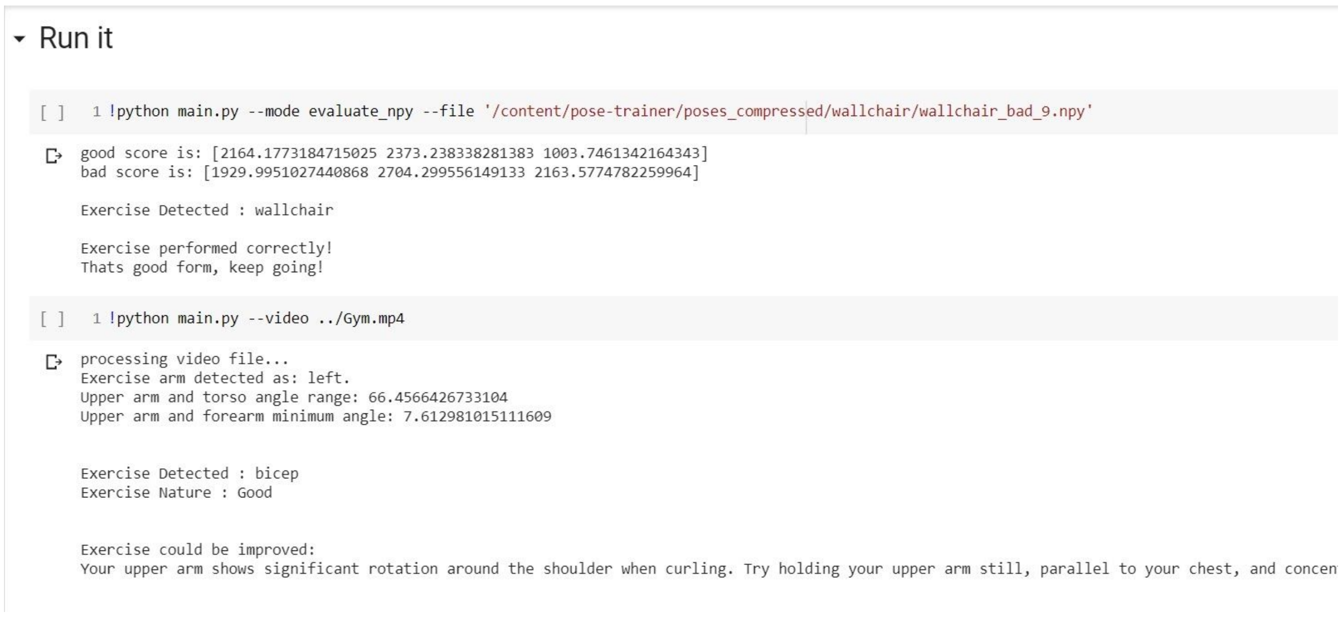

Speed (km/hr) : 23.881 ---- Angle : [46.925] ---- Frame No : 5

Fig 6.3.1 Speed Calculation

The speed in the above video was not proper so I had to modify certain values in the code. After modifying the values the new speed and angle values are 23Km/hr and 46° as shown in the image.

6.4 Pose Estimation and Angles Generation

The results of pose estimation and angle generation are shown below.



Run it

```
!python main.py --mode evaluate_npy --file '/content/pose-trainer/poses_compressed/wallchair/wallchair_bad_9.npy'
```

```
good score is: [2164.1773184715025 2373.238338281383 1003.7461342164343]
bad score is: [1929.9951027440868 2704.299556149133 2163.5774782259964]

Exercise Detected : wallchair

Exercise performed correctly!
Thats good form, keep going!
```

```
!python main.py --video ../Gym.mp4
```

```
processing video file...
Exercise arm detected as: left.
Upper arm and torso angle range: 66.4566426733104
Upper arm and forearm minimum angle: 7.612981015111609


Exercise Detected : bicep
Exercise Nature : Good


Exercise could be improved:
Your upper arm shows significant rotation around the shoulder when curling. Try holding your upper arm still, parallel to your chest, and concen
```

Fig 6.4.1 Angles Generation

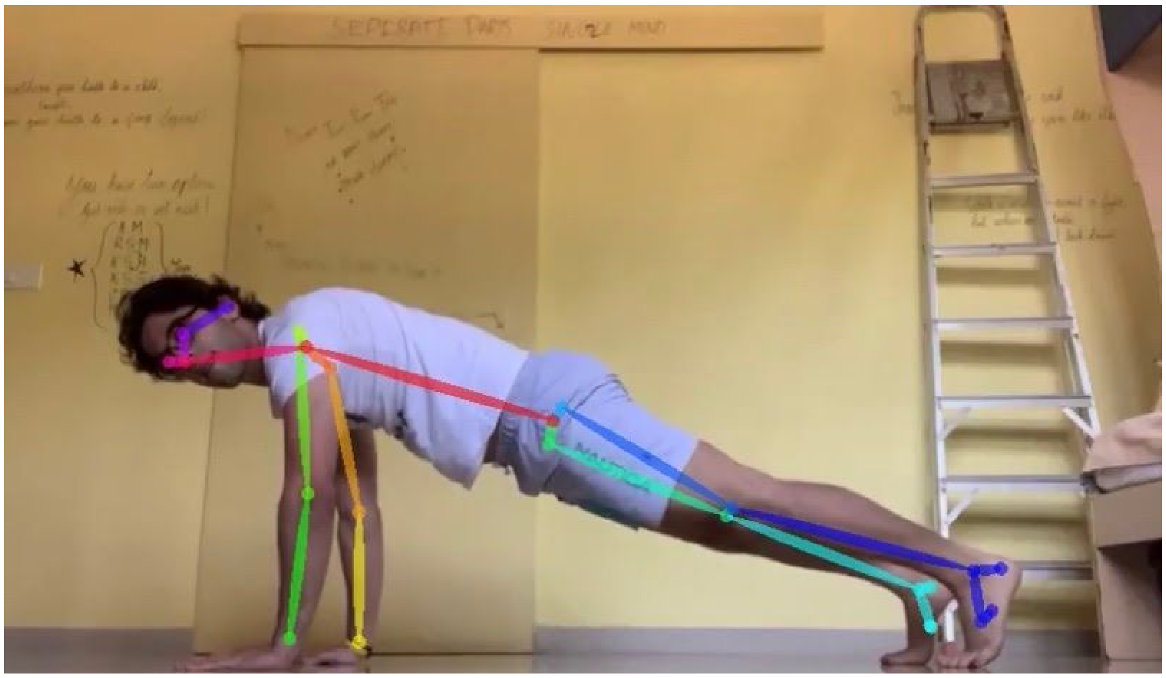


Fig 6.4.2 Pose Estimation

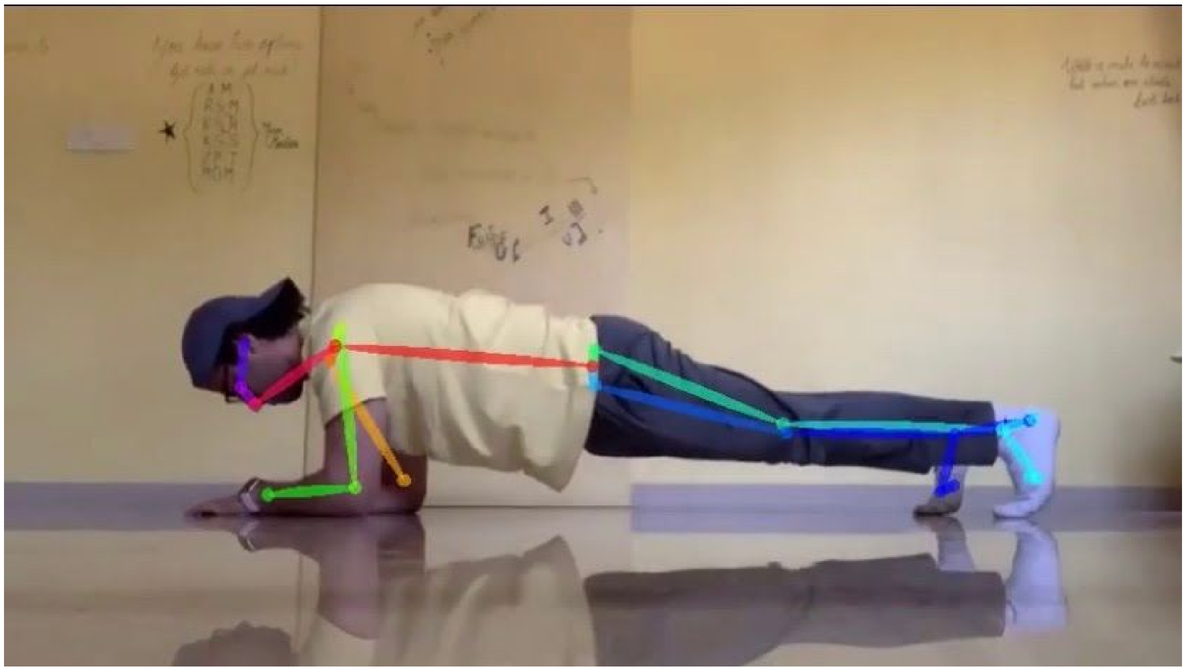
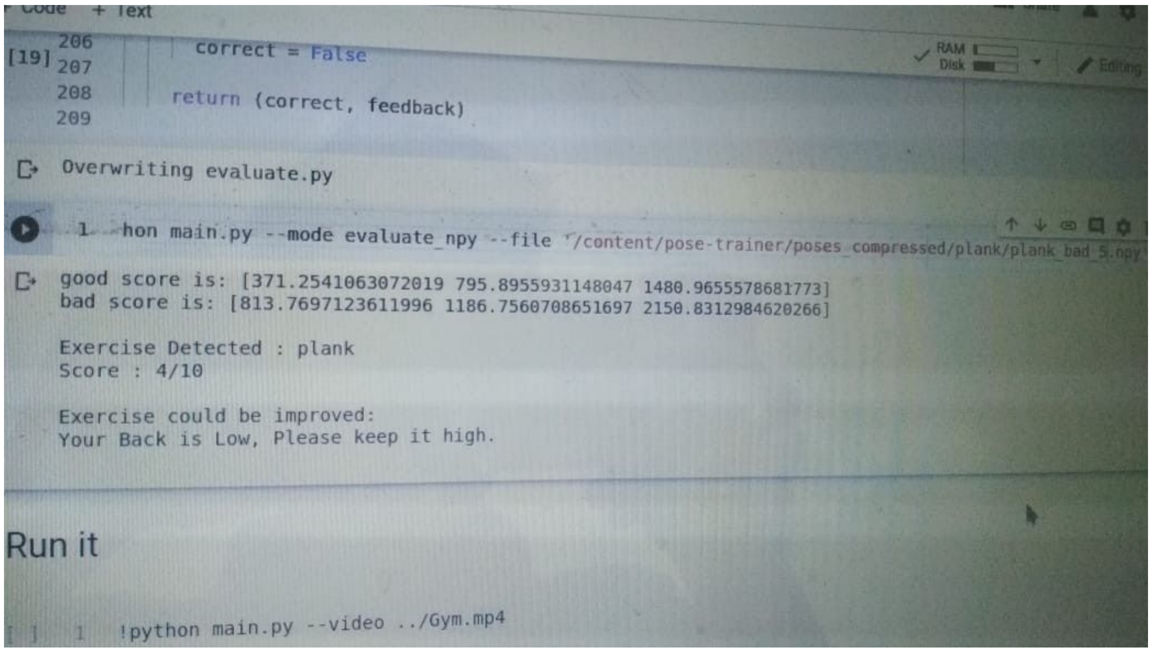


Fig 6.4.3 Pose Estimation

6.5 Classification and Correction of Exercise

The results of Classification and Correction of Exercise are shown below.



```
[19]
        correct = False

    return (correct, feedback)
```

```
Overwriting evaluate.py
```

```
hon main.py --mode evaluate_npy --file '/content/pose-trainer/poses_compressed/plank/plank_bad_5.npy'
```

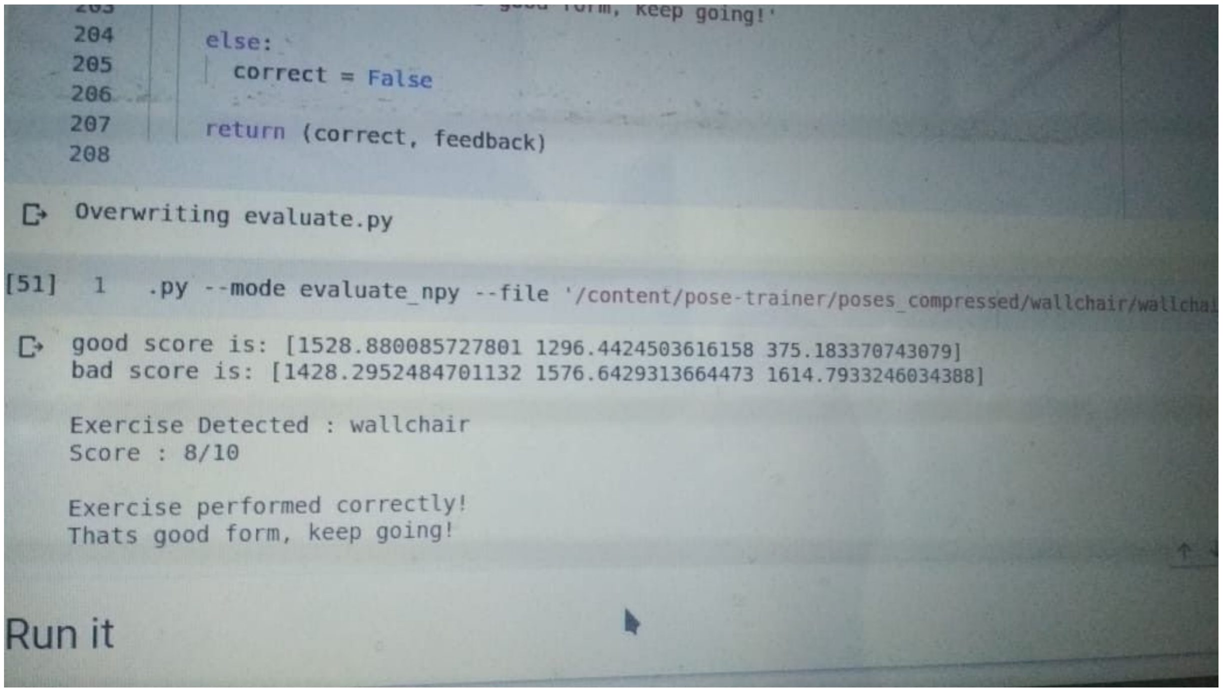
```
good score is: [371.2541063072019 795.8955931148047 1480.9655578681773]
bad score is: [813.7697123611996 1186.7560708651697 2150.8312984620266]

Exercise Detected : plank
Score : 4/10

Exercise could be improved:
Your Back is Low, Please keep it high.
```

Run it

```
!python main.py --video ../Gym.mp4
```

Fig 6.5.1 Classification of Exercises

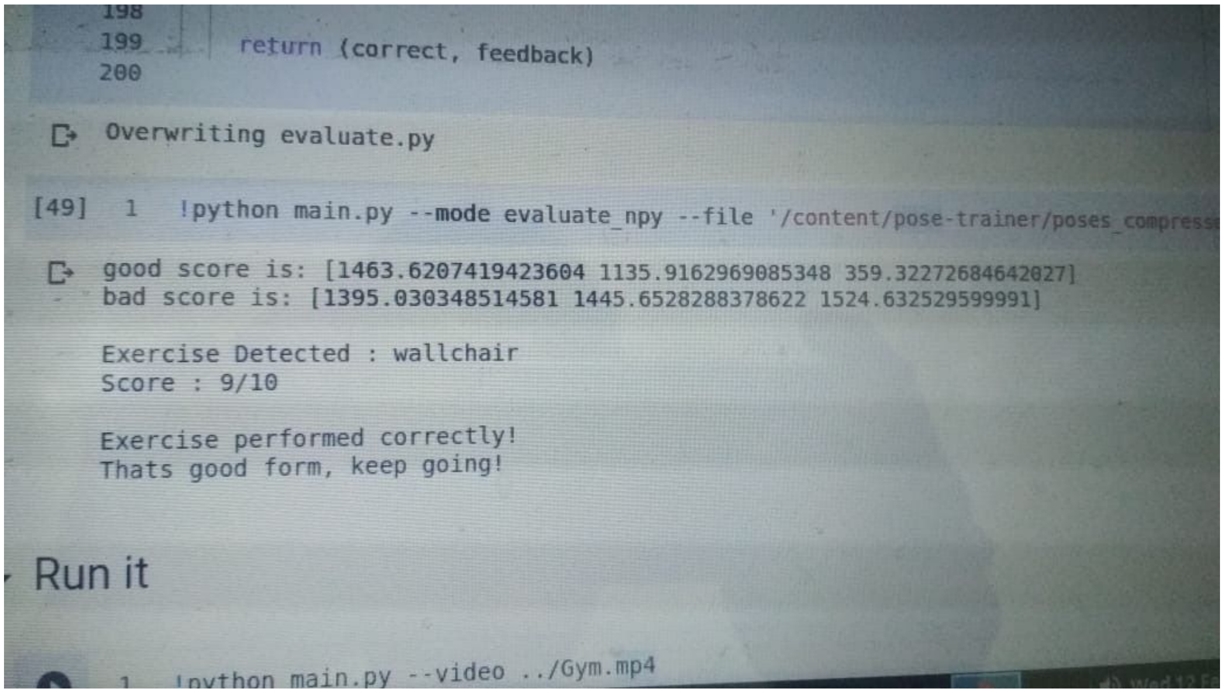
Here I am classifying and correcting the exercises based on the scores given to the exercise. I am also scaling the good and bad scores of exercise out of 10. Score for each exercise can range between 0 and 10 (both included).



```
    else:
      correct = False

    return (correct, feedback)

Overwriting evaluate.py

[51]  .py --mode evaluate_npy --file '/content/pose-trainer/poses_compressed/wallchair/wallchai

good score is: [1528.880085727801 1296.4424503616158 375.183370743079]
bad score is: [1428.2952484701132 1576.6429313664473 1614.7933246034388]

Exercise Detected : wallchair
Score : 8/10

Exercise performed correctly!
Thats good form, keep going!
```

Run it

Fig 6.5.2 Exercise Correction



```
    return (correct, feedback)

Overwriting evaluate.py

[49]  !python main.py --mode evaluate_npy --file '/content/pose-trainer/poses_compress

good score is: [1463.6207419423604 1135.9162969085348 359.32272684642027]
bad score is: [1395.030348514581 1445.6528288378622 1524.632529599991]

Exercise Detected : wallchair
Score : 9/10

Exercise performed correctly!
Thats good form, keep going!
```

Run it

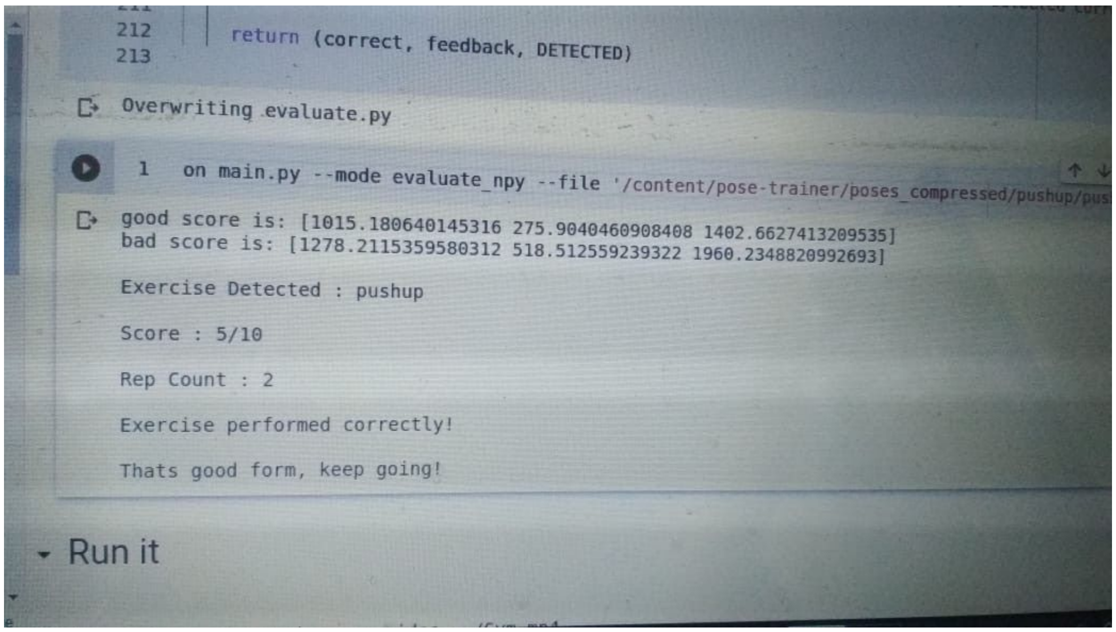
```
!python main.py --video ../Gym.mp4
```

Fig 6.5.3 Exercise Scoring

If the exercise is being performed in a correct and proper way then the score of that exercise should be above 5 out of 10 that is between 5 and 10 (including 10). If the exercise is not performed in a correct way and needs proper feedback to improve, the score of that exercise should be below 5 out of 10 that is between 0 and 5 (including 0).

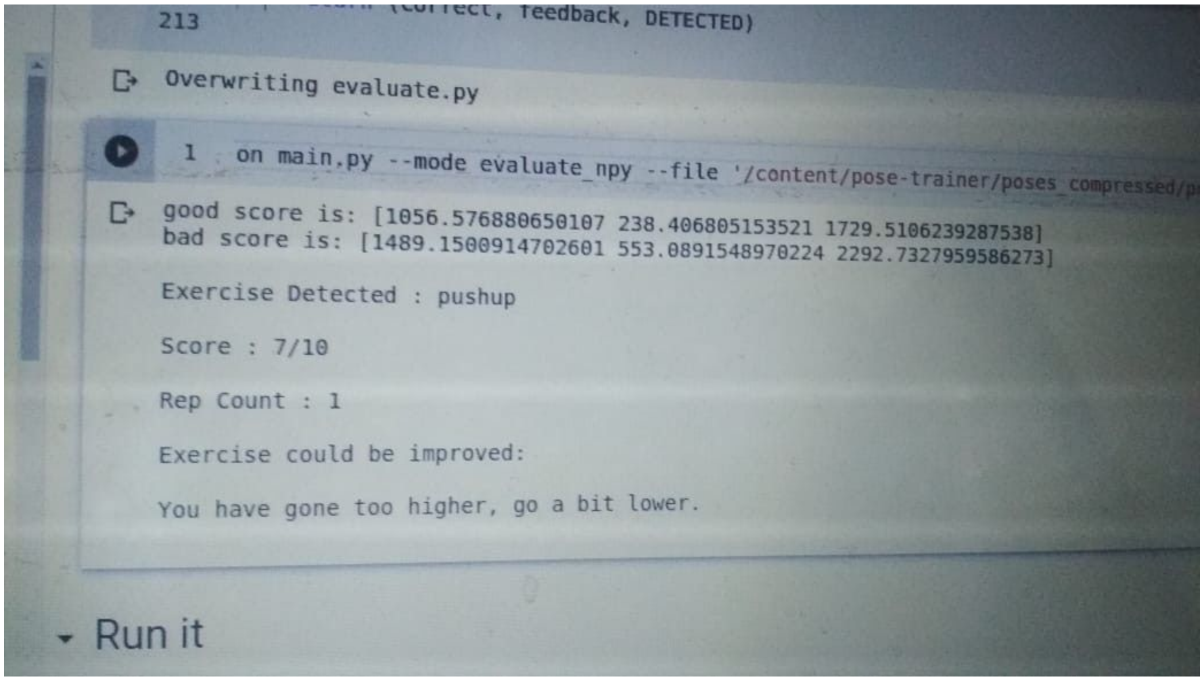
6.6 Rep Count Analysis

The results of Repetition Count Analysis are shown below.



```
    return (correct, feedback, DETECTED)

Overwriting evaluate.py

on main.py --mode evaluate_npy --file '/content/pose-trainer/poses_compressed/pushup/pus

good score is: [1015.180640145316 275.9040460908408 1402.6627413209535]
bad score is: [1278.2115359580312 518.512559239322 1960.2348820992693]

Exercise Detected : pushup

Score : 5/10

Rep Count : 2

Exercise performed correctly!

Thats good form, keep going!
```

Fig 6.6.1 Repetition Count



```
213            (correct, feedback, DETECTED)

Overwriting evaluate.py

1   on main.py --mode evaluate_npy --file '/content/pose-trainer/poses_compressed/p

good score is: [1056.576880650107 238.406805153521 1729.5106239287538]
bad score is: [1489.1500914702601 553.0891548970224 2292.7327959586273]

Exercise Detected : pushup

Score : 7/10

Rep Count : 1

Exercise could be improved:

You have gone too higher, go a bit lower.
```

Fig 6.6.2 Rep Count Analysis

The figure shows the output of repetition count of various exercises namely pushup, plank, wall chair etc. I have implemented many exercises and I am getting the output of the repetition count of all those exercises.

I have also provided the feedback to all the exercises performed based on the angles. If the required angle goes above than the maximum range or goes below than the minimum range then I am providing the necessary feedback to the exercise.

Chapter 7

Conclusion

For the project the Automated Exercise Trainer I developed a model which detects the exercise performed and provides necessary feedback if the exercise needs to be improved. Hence I performed classification and correction of exercises using machine learning and deep learning techniques. Then I scored each exercise performed and assigned good and bad scores to every exercise in the form of an array. After that I scaled the scores of an exercise in the range 0 to 10. Hence I scored each exercise out of 10. I also extracted several body angles from the exercises. After that, I performed repetition count analysis that is I counted the reps from the exercise performed. After finishing these all, I need to implement this python code in Android Studio hence I used Chaquopy Library to implement my python code in Android Studio. At last, I got the output of my code in the Android Studio console.

In the project Game Statistics, I developed a model to get the most out of a mobile phone application without using any external sensors. I started with analysing the object involved in the game like shuttle in badminton and ball in tennis. Then I trained a YOLO model to detect objects and for tracking the trajectory of the object in a video. This helped me exploit a few characteristics of the image as I passed these coordinates available to the depth estimation model, the speed calculation model, the angle calculation mode, the point of hit calculation model, the bounce point calculation model and so on. Each of the tasks had their own dedicated code and methodology to calculate the parameters whose respective outputs are shown in the results section. The homography along with line detector and corner detector is used to give the graphical representation of the point of bounce. 80% of the dataset and videos used for training were created by me using my phone camera. Further I merged the various modules of the code into one.