STUDENTS' TECHNICAL COUNCIL

Project:

Hand Cricket

Brief Description:

Playing Hand Cricket with a computer, with gesture detection.

Ideation Process:

1) **Rules of the Game**- Detailed discussions regarding the rules of the game.

    a. The game will offer the players up to six hand gestures (1to 6) to play the game.
    b. As of now the game will only be single player game in which player plays against computer.
    c. Computer will generate gestures using random function and whenever that value matches with player gesture, then computer /player will be out respectively.

Hand Gesture Detection:

Studying various online examples and hand detection repositories to come up with the best hand detection method.
Choices between - OpenCV, Yolo, and CNN.

1) OpenCV- Less error-prone and a lot faster, but difficult to configure.

2) Yolo - Training of data requires a lot of time as it requires labeling of each image.
3) CNN - Too slow, and the response time is way ahead of other methods. Not feasible for this project.

Work on OpenCV:

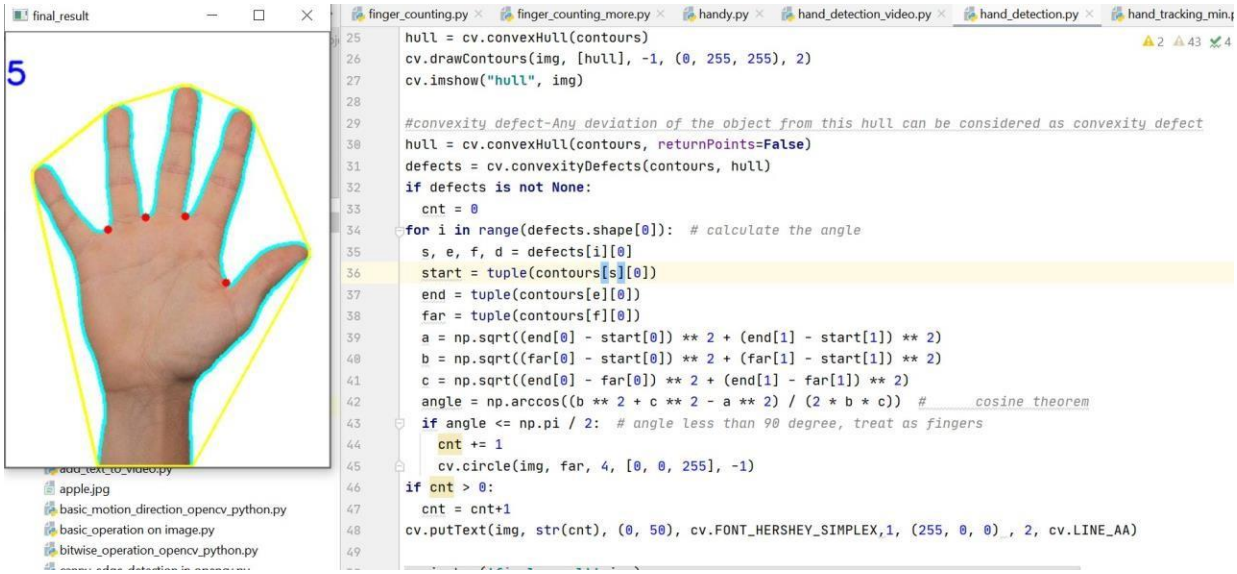
We start with a basic medium article, "**Hand Detection and Finger Counting Using OpenCV-Python**."

Reference:

https://medium.com/analytics-vidhya/hand-detection-and-finger-counting-us ing-opencv-python-5b594704eb08

We used simple OpenCV techniques like Skin Masking, Contours, Convex Hull, and Convexity defects and rules of cosine theorem inthis algorithm.

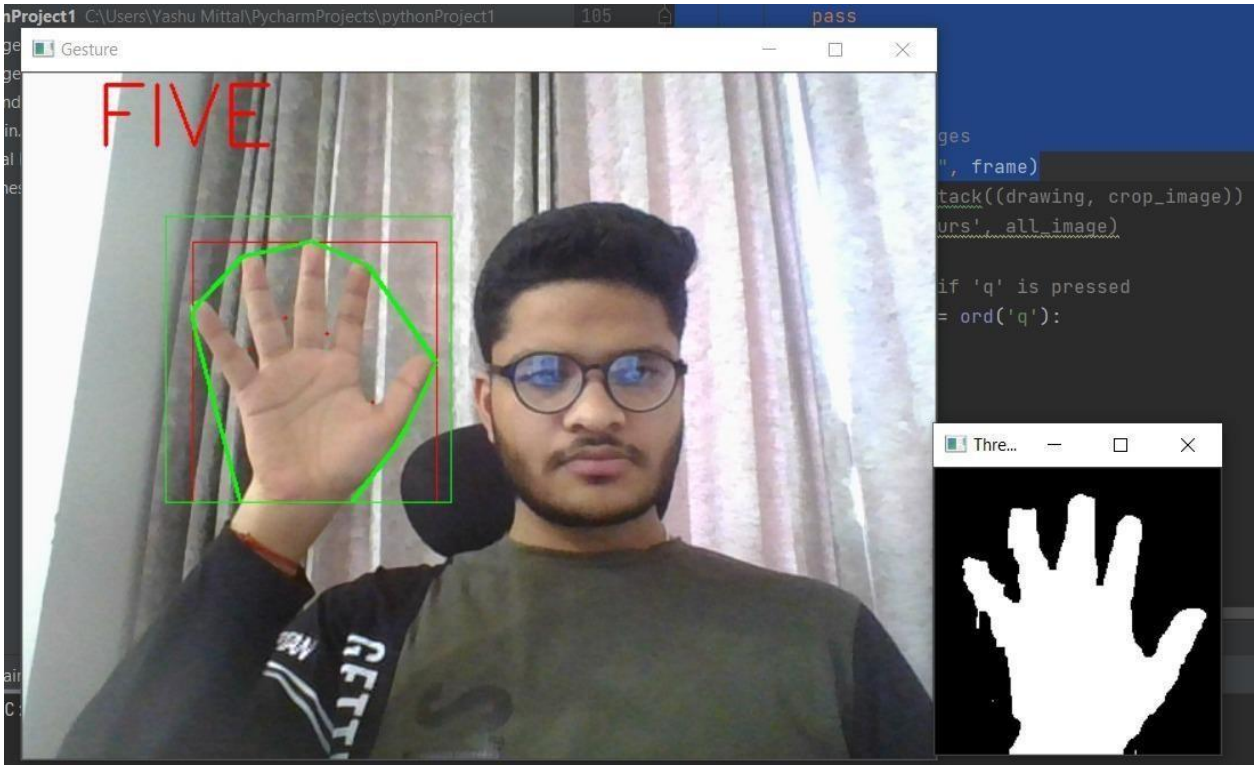
And we got the results as shown below:

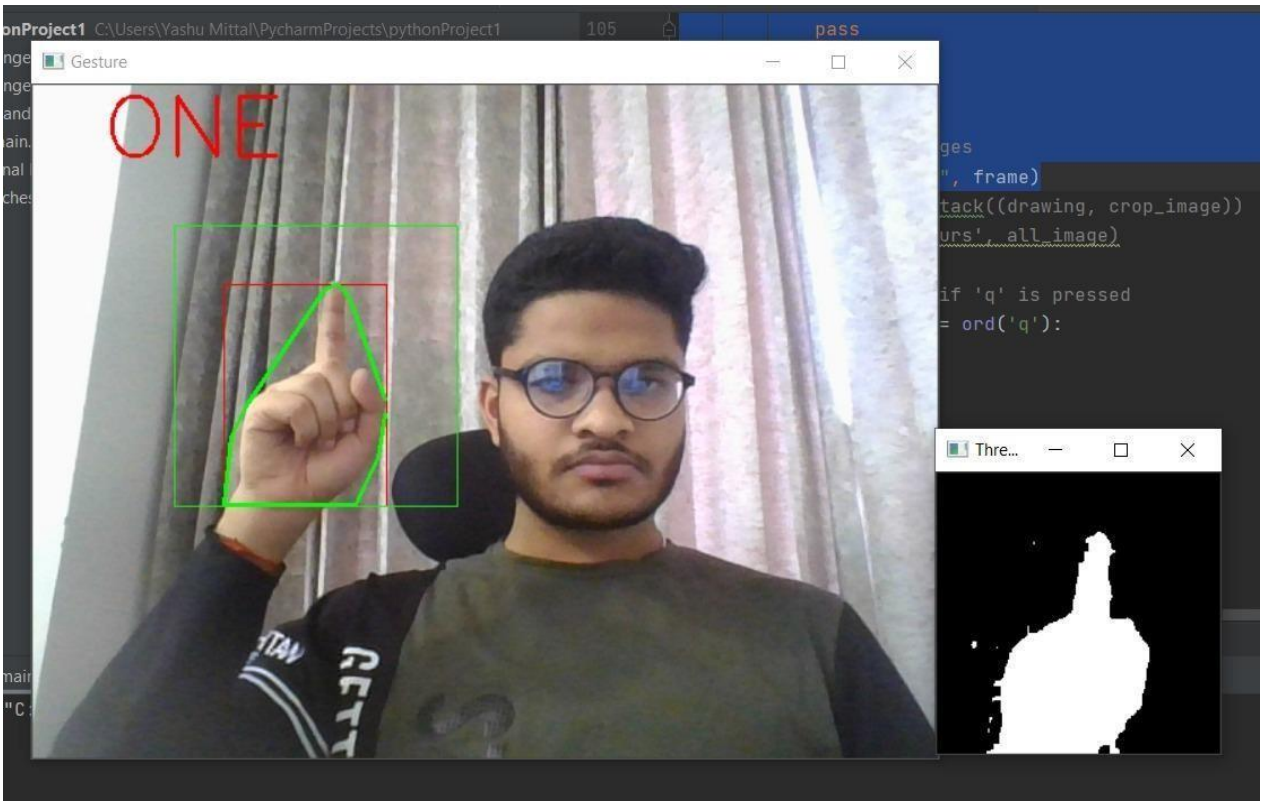


Code available at:

https://github.com/ariesiitr/Hand-Cricket/blob/shrashtika/hand_detection.py

After that, we implemented this algorithm on video frames as our project needs to work in real-time detection.


Gesture
FIVE
Thre...
pass
ges
", frame)
tack((drawing, crop_image))
urs', all_image)
if 'q' is pressed
= ord('q'):


Gesture
ONE
Thre...
pass
ges
", frame)
tack((drawing, crop_image))
urs', all_image)
if 'q' is pressed
= ord('q'):

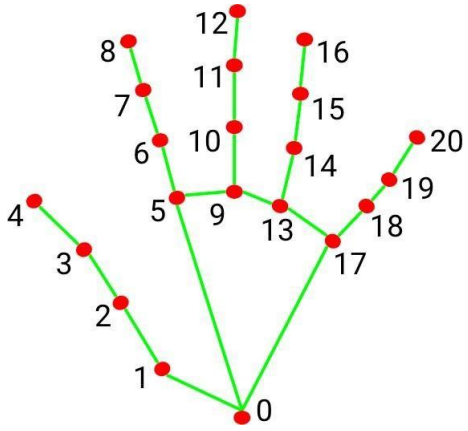
Code available at:
https://github.com/ariesiitr/Hand-Cricket/blob/yatin/main.py

In this code, we found that detection is good, but fluctuations are very high then; we tried to find some method to reduce this fluctuation or anew method with very fewer fluctuations.

In a new method, we use ***MediaPipe***.
MediaPipe offers customizable Python solutions as a prebuilt Python package on PyPI, which can be installed simply with ***pip install mediapipe***. It also provides tools for users to build their solutions.

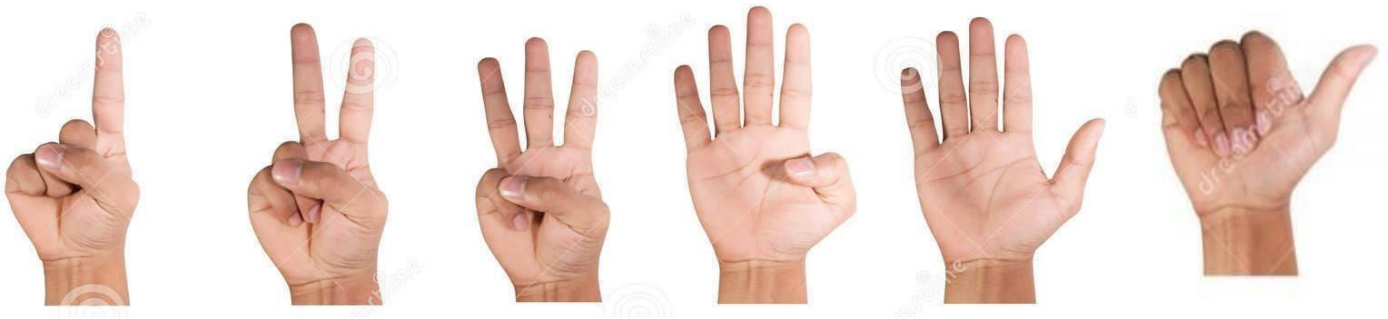
Firstly, for palm detection, we use a pre-built ***Palm detection model*** inMediaPipe. MediaPipe Hands utilizes an ML pipeline consisting of multiple models working together. A palm detection model that operates on the full image and returns an oriented hand bounding box.A hand landmark model operates on the cropped image region defined by the palm detector and returns high-fidelity 3D hand keypoints.

After the palm detection over the whole image, our subsequent ***Handlandmark model*** performs precise keypoint localization of ***21 3D hand-knuckle coordinates*** inside the detected hand regions via regression, that is direct coordinate prediction. The model learns a consistent internal hand pose representation and is robust even to partially visible hands and self-occlusions.



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

Link for the reference:
https://google.github.io/mediapipe/solutions/hands
To implement this algorithm first, we create a class "***handDetector***"available at:
https://github.com/ariesiitr/Hand-Cricket/blob/shrashtika/HandTrackingModule.py
We use this class in our actual model.
Using MediaPipe, the accuracy rate is high and better than the previous algorithm.
First, we write a code to simply count fingers 0 to 5 instead of detecting a particular hand pose.

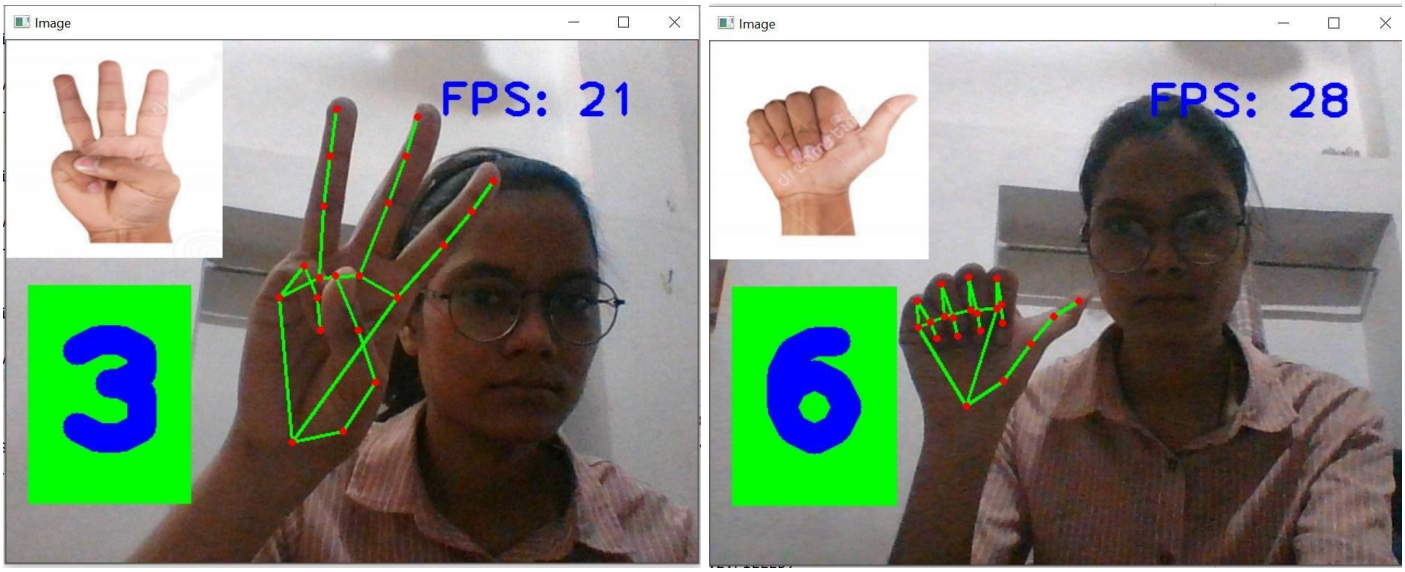


For detecting particular six hand poses shown in the above snap to make our model more effective, we write unique code lines for eachparticular hand pose.

Algorithm used:

To detect the particular hand pose of the ***right hand*** we write unique code for each hand pose, the algorithm used for 4 fingers is up-down.If the top point of any of the fingers is above the bottom point of fingerthat means the finger is open otherwise close. For thumb we used theleft-right algorithm. If the thumb of the right hand is right side of the screen then it would be considered as open otherwise close.

Results:

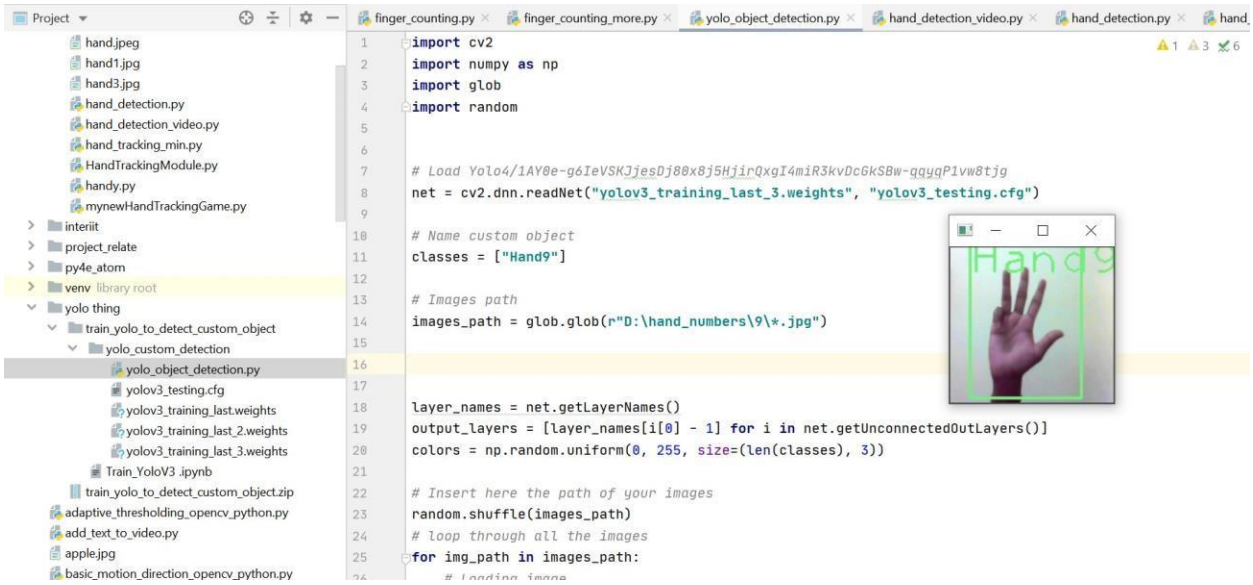


Code available at:
https://github.com/ariesiitr/Hand-Cricket/blob/shrashtika/finger_counting_more_1to6.py
Till now, we have found this code is suitable for our model.

YOLO Work and why we are selecting OpenCV over YOLO:

Reference:
https://pysource.com/2020/04/02/train-yolo-to-detect-a-custom-object-online-with-free-gpu/
We needed to prepare an image dataset of 10 different hand gestures, and for each hand gesture minimum of 350 images. And then specify where the custom object is located on the specific image.
For this operation, we used an external software: ***LabelImg*.**
We need to specify the custom objects in each specific image i.e., in 350*10 images.
To train the image dataset, we used the free server offered by google colab(Free GPU). For this, we used a notebook named ***"Train_YoloV3.ipynb"*.**

We are required to complete at least 1000 iterations of training. We used a pre-built python project to test your model with OpenCV.



Code available at:
https://github.com/ariesiitr/Hand-Cricket/blob/shrashtika/yolo_object_detection.py
This model was working well to some extent, but the problem is that it is very time-consuming, like the specification of the custom object using Labeling and training of dataset on google colab. Due to this, we switch to OpenCV.

Progress so far:

Successfully implemented OpenCV code for gesture recognition of 1 through 6. Currently, the work has been done for the right hand. The same can be done for the left hand, with some minor tweaks in the code.
The problem of background detection and uneven background was also resolved. Now the gestures can be recognised in a variety of different backgrounds.