MINI PROJECT - FINGER COUNT

PROBLEM STATEMENT:

- It's easy to count our fingers through our eyes but it's difficult to count the fingers by processing and analyzing the video.
- The idea is to solve the problem of counting the number of fingers through the live video by accessing the camera.

AIM:

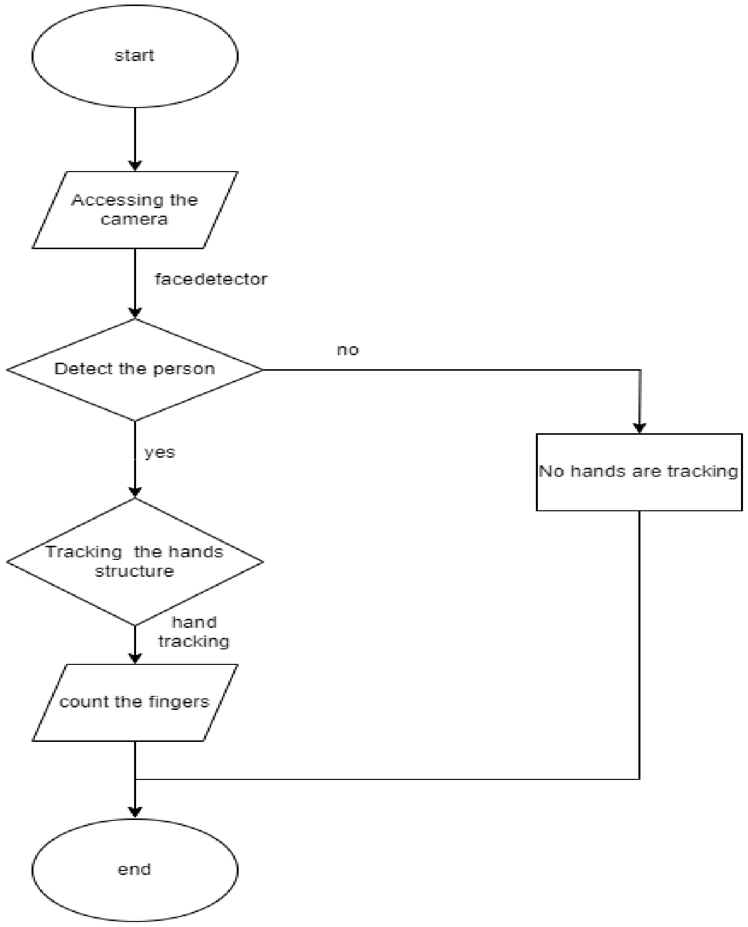
To build a mini project to count the number of fingers in a hand by analyzing the video.

INTRODUCTION:

1.This project is all about counting the number of fingers and also it finds whether the hand is right or left.Project calculates the number of fingers in one hand only.

2.If we show two hands in the same frame at the same time ,it produces the output as the maximum fingers present in the hands(ranges from 0 to 5).

WORKFLOW:



MODULE AND FUNCTION DESCRIPTION:

- **cv2:** OpenCV (Open Source Computer Vision) is a library of programming functions mainly aimed at real-time computer vision. It is used for image and video processing. cv2 is the Python interface for OpenCV.
- **cvzone.FaceDetectionModule**: This is a module from the cvzone library that provides pre-trained models for face detection in images and videos.
- **cvzone.HandTrackingModule:** This is a module from the cvzone library that provides pre-trained models for hand detection and tracking in images and videos.
- **os:** This is a module that provides a way of using operating system dependent functionality like reading or writing to the file system, and handling paths.
- **cvzone:** This is a library that provides computer vision utilities such as FPS counter, color detection, etc.
- **cvzone.FPS():** This is a class from the cvzone library that helps to calculate and display the frame rate of the video.
- **cap = cv2.VideoCapture(0)**: This function creates a VideoCapture object to capture video from the default camera (webcam).
- **cap.set(10,200):** This function sets the brightness of the video captured by the camera.
- **hd = HandDetector(detectionCon=0.6):** This function creates a HandDetector object with a detection confidence of 0.6, which is used for hand detection and tracking.
- **os.listdir(folderpath):** This function returns a list of all files and directories in the specified folderpath.
- **cv2.imread():** This function reads an image from the specified file path.
- **_, img = cap.read()**: This function captures the video frame-by-frame and returns a boolean (True if the frame is read correctly) and the frame itself.
- **hd.findHands(img)**: This function detects and tracks hands in the specified img frame using the HandDetector object.
- **hd.fingersUp(lefthand):** This function counts the number of fingers that are extended in the detected lefthand.
- **cv2.rectangle(img, (0, 200), (170, 425), (0, 0, 255), cv2.FILLED):** This function draws a filled rectangle on the specified img frame with the specified color and coordinates.
- **cv2.putText(img, str(totalfingers), (45, 375), cv2.FONT_HERSHEY_PLAIN, 9, (255, 0, 0), 24):** This function writes the count of extended fingers on the specified img frame at the specified position and with the specified color and font.
- **cv2.imshow('FRAME',img):** This function displays the specified img frame in a window with the specified name.

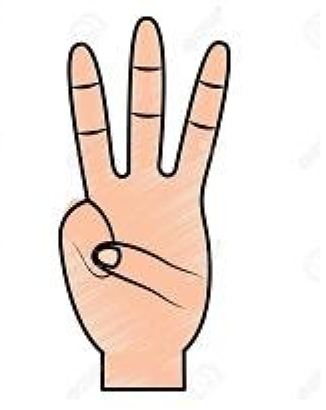
- **cv2.waitKey(1):** This function waits for a specified delay for a keyboard event. If a key is pressed, it returns the ASCII value of the key.
- **cap.release()**: This function releases the camera resources.
- **cv2.destroyAllWindows():** This function destroys all the windows created by OpenCV

SOFTWARE AND HARDWARE REQUIREMENTS:
Processor : Intel(R) Core(TM) i5-6300U CPU @ 2.40GHz   2.50 GHz.
RAM : 8GB.
Software : Anaconda Navigator.

SOURCE CODE:

```
import cv2
from cvzone.FaceDetectionModule import FaceDetector
from cvzone.HandTrackingModule import HandDetector
import os
import cvzone
fps = cvzone.FPS()

cap = cv2.VideoCapture(0)
cap.set(10,200)

hd = HandDetector(detectionCon=0.6)


overlaylist=[]
folderpath = 'C://Users//ADMIN//Downloads//Python-codes-main//finger up counter//Fingers'
list = os.listdir(folderpath)
print(folderpath)
for imgpath in list:
    image = cv2.imread(f'{folderpath}/{imgpath}')
    overlaylist.append(image)



while True:
    _, img = cap.read()
    fps.update(img,pos=(490,40),scale=2,color=(0,0,255))
    hand,imgs = hd.findHands(img)
    if hand:
        lefthand = hand[0]
        bbox = lefthand["bbox"]
        lmlist = lefthand['lmList']
        handtype = lefthand['type']
        fingersup = hd.fingersUp(lefthand)
        totalfingers = fingersup.count(1)
        h, w, c = overlaylist[totalfingers - 1].shape
        img[0:h, 0:w] = overlaylist[totalfingers - 1]
        cv2.rectangle(img, (0, 200), (170, 425), (0, 0, 255), cv2.FILLED)
        cv2.putText(img, str(totalfingers), (45, 375), cv2.FONT_HERSHEY_PLAIN, 9, (255, 0, 0),
24)
        //print(totalfingers)
    cv2.imshow('FRAME',img)
    key = cv2.waitKey(1)
    if key == 27:
        break
cap.release()
cv2.destroyAllWindows()
```

FINGER FOLDER CONTAINS THIS SIX IMAGES:



OUTPUT:

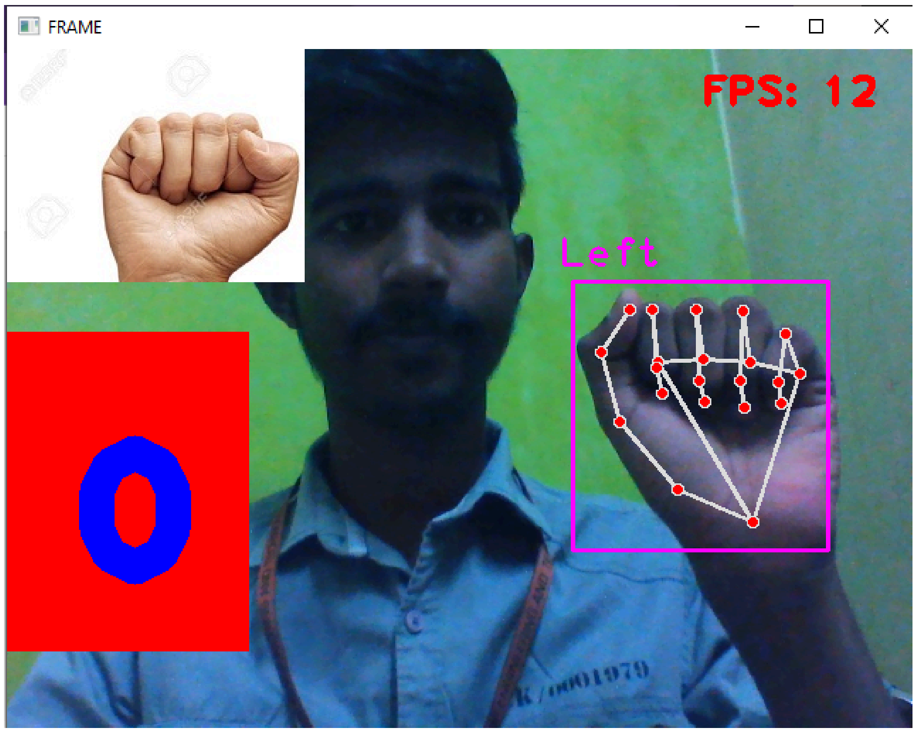
C://Users//ADMIN//Downloads//Python-codes-main//finger up counter//Fingers

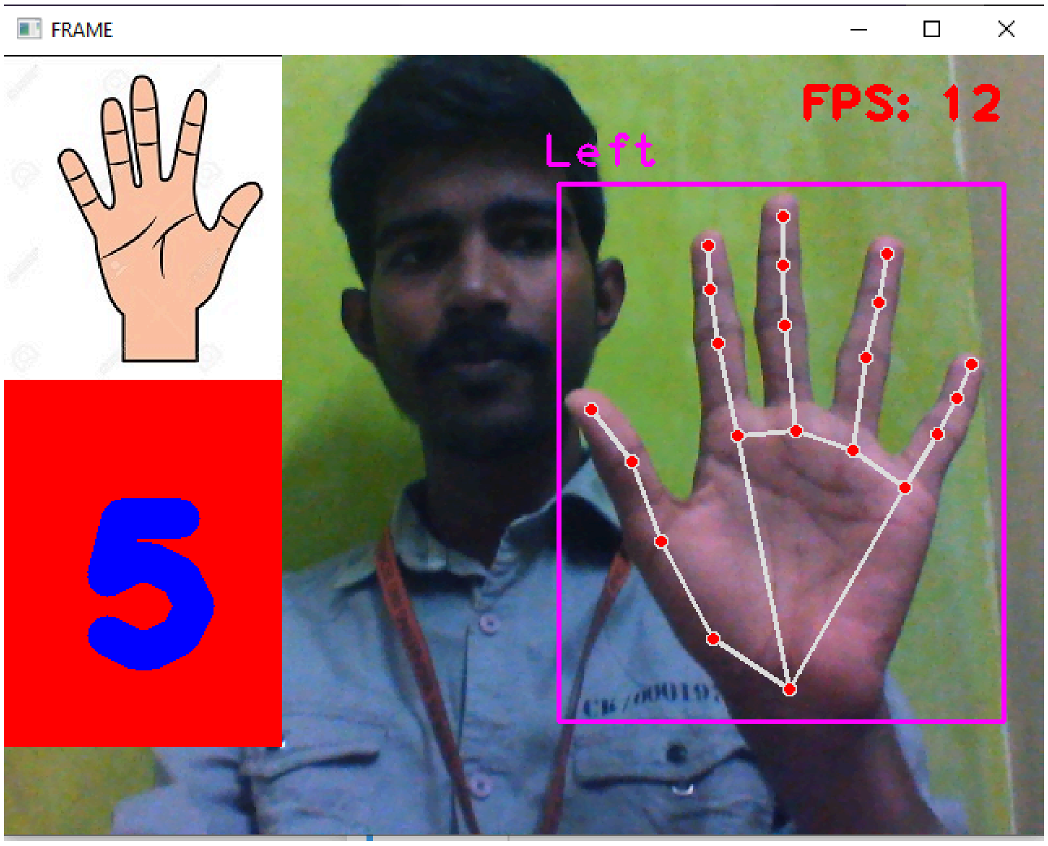
TO STOP THE FRAME ,WE HAVE TO STOP THE EXECUTING PROGRAM.

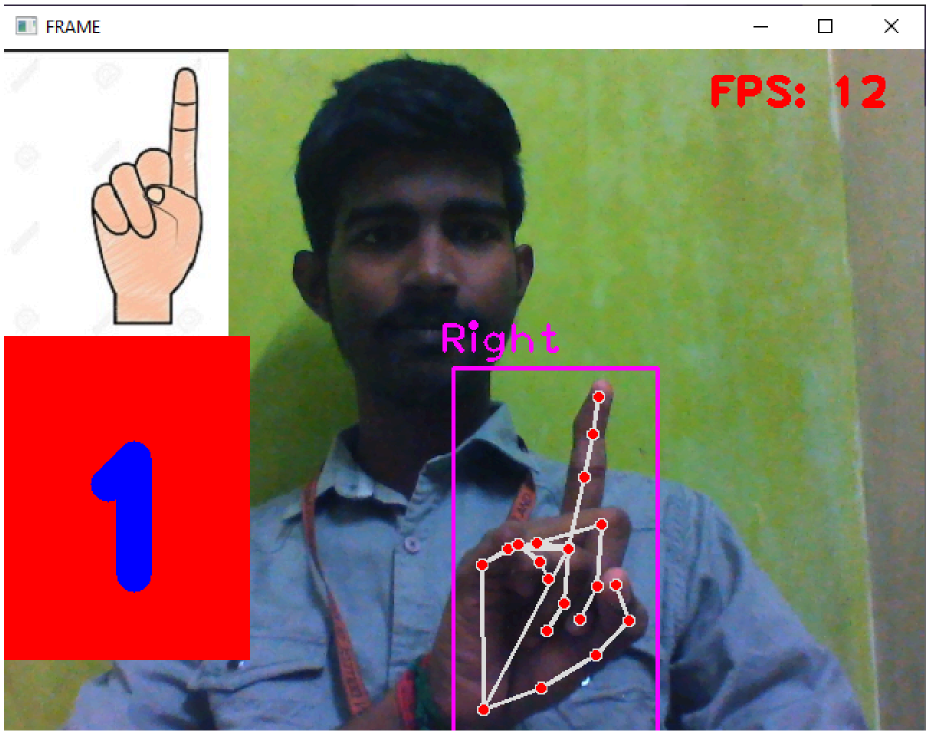
```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
Input In [1], in <cell line: 25>()
     25 _, img = cap.read()
     26 fps.update(img,pos=(490,40),scale=2,color=(0,0,255))
---> 27 hand,imgs = hd.findHands(img)
     28 if hand:
     29     lefthand = hand[0]

File ~\anaconda3\lib\site-packages\cvzone\HandTrackingModule.py:49, in
HandDetector.findHands(self, img, draw, flipType)
     42 """
     43 Finds hands in a BGR image.
     44 :param img: Image to find the hands in.
     45 :param draw: Flag to draw the output on the image.
     46 :return: Image with or without drawings
     47 """
     48 imgRGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
---> 49 self.results = self.hands.process(imgRGB)
     50 allHands = []
     51 h, w, c = img.shape

File
~\AppData\Roaming\Python\Python39\site-packages\mediapipe\python\solutions\hands.py:
153, in Hands.process(self, image)
    132 def process(self, image: np.ndarray) -> NamedTuple:
    133   """Processes an RGB image and returns the hand landmarks and handedness of each
detected hand.
    134
    135   Args:
   (...)
    150       right hand) of the detected hand.
    151   """
--> 153   return super().process(input_data={'image': image})

File
~\AppData\Roaming\Python\Python39\site-packages\mediapipe\python\solution_base.py:3
65, in SolutionBase.process(self, input_data)
    359   else:
```

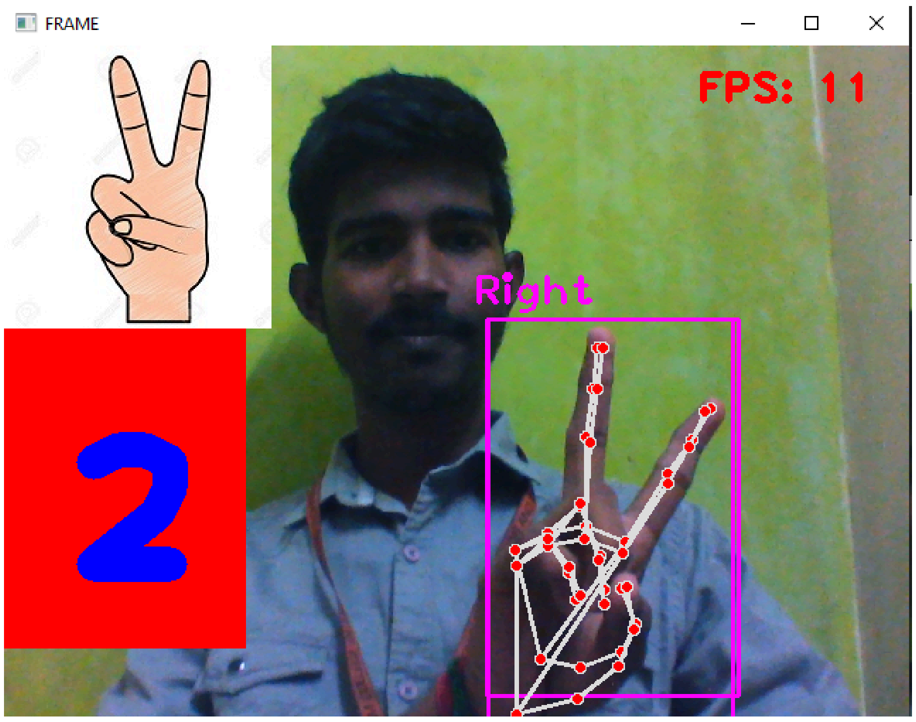
```
    360     self._graph.add_packet_to_input_stream(
    361         stream=stream_name,
    362         packet=self._make_packet(input_stream_type,
    363                                  data).at(self._simulated_timestamp))
--> 365 self._graph.wait_until_idle()
    366 # Create a NamedTuple object where the field names are mapping to the graph
    367 # output stream names.
    368 solution_outputs = collections.namedtuple(
    369     'SolutionOutputs', self._output_stream_type_info.keys())

KeyboardInterrupt:
```

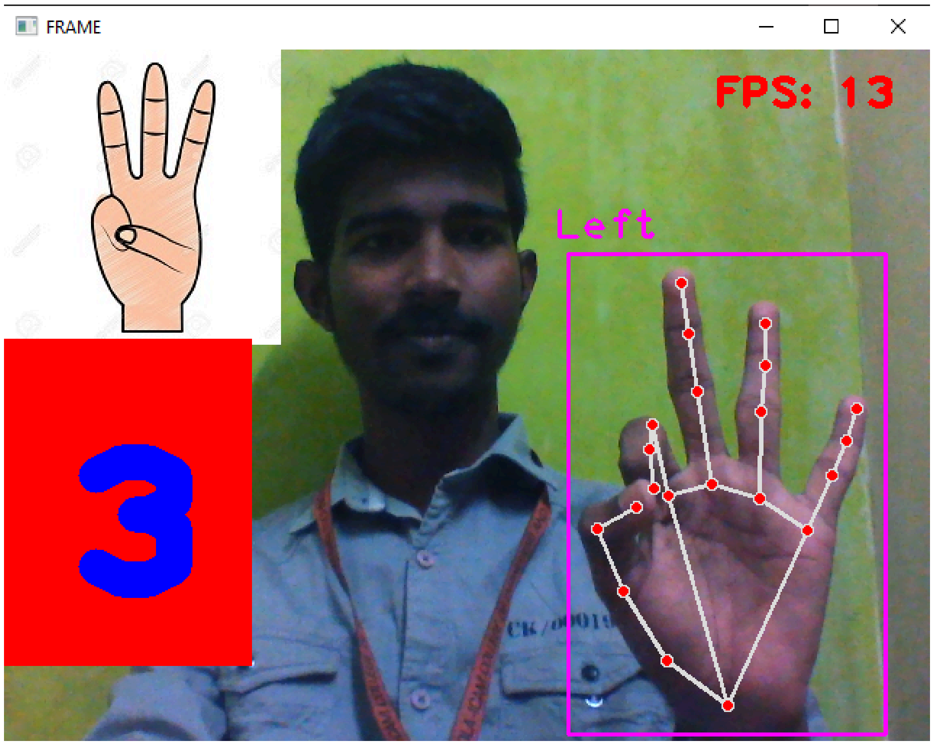
SNAPSHOT:




FRAME
FPS: 12
Left
5


FRAME
FPS: 12
Right
1


FRAME
FPS: 11
Right
2



INFERENCE:

By using the cvzone package ,we can generate up to 13 fps(FRAMES PER SECONDS) and it also classifies the hand(right or left).

RESULT:

Thus the mini-project to count the number of fingers by accessing the live video was executed successfully.