AI GESTURE PAINTER

ABOUT

This project is an AI gesture painter or type writer that recognizes and prints the written text by our index finger. This project is made on python using computer vision (OpenCV).

HOW TO USE THE PROGRAM?

Given below are the various modes:

1) **Writing mode** (condition: only index finger up): the tip of index finger is considered for drawing.
2) **Pause mode** (condition: only index and middle finger up): the program stops to draw. This mode is used to provide spaces between characters and to hover around the canvas without drawing.
3) **Detection mode** (condition: index, middle and ring finger up): it is used to detect the written text after the user is done writing.
4) **Clear mode** (condition: all four fingers up): it is used for clearing the canvas by removing the text written previously.

MODULES USED

1. **Mediapipe :-**There are many machine learning solutions inside mediapipe that includes face detection, face mesh, iris detection, hand detection. We used Hands inside mediapipe. 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 that operates on the cropped image region defined by the palm detector and returns high-fidelity 3D hand keypoints.

2. **Pytesseract :-** Pytesseract is an optical character recognition (OCR) tool for python. That is, it will recognize and “read” the text embedded in images or

videos. Tesseract finds templates in pixels, letters, words and sentences. It uses two-step approach that calls adaptive recognition. It requires one data stage for character recognition, then the second stage to fulfil any letters, it wasn't insured in, by letters that can match the word or sentence context.

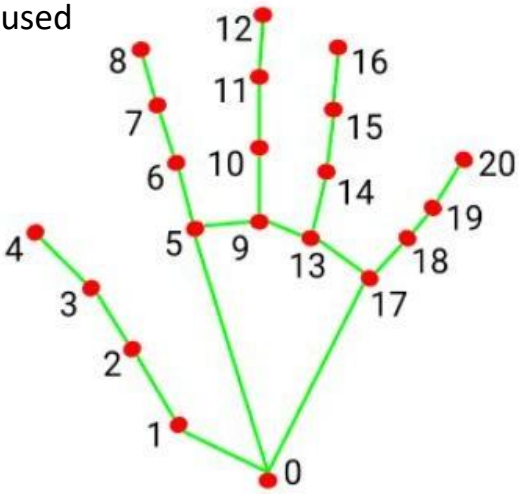
3. **Numpy :-** NumPy is a Python library used for working with arrays.

WORKING

We made 2 files one is the hand tracker module file and the other one is the main program file.

Hand tracker module – The module was made using object oriented programming, inside this module mediapipe hands was used to detect hands and to make labels around hand.

The tips of fingers respectively are 4,8,12,16 and 20 further in the class hand detector a find position function was made to get (x,y) coordinates of all points detected.Now for detection of how many fingers are up



Another function was defined under class that finds how many fingers are up. For example index finger up is identified when y coordinate of 8 is more than that of 7, and for thumb when x coordinate of 4 is less than that of 3. Thumb is finger[0], and rest of fingers are finger[i] (i={1,2,3,4}).

Main file - The read camera feed was flipped so that it would be comfortable for writing. (x,y) coordinates of index finger were taken for drawing and when both index and middle finger were up then x, y coordinates were taken as 0.

For drawing a canvas was created from zeros of numpy array as drawing is not reflected on camera feed image, to draw on canvas cv2.line was used to draw between to 2 points inside loop. To reflect the drawing on camera feed image bitwise and and bitwise or were used between camera feed image and canvas.

Bitwise and to reflect the drawing part on cam image at same position in black colour, and bitwise or to colour the drawing part with the actual drawing colour.

For detection part Pytesseract was used when index, middle and ring finger are up, for pytesseract to run location of tesseract exe file is required to run OCR detection. As an exe file runs simultaneously for detection so it lags a little while detection. Pytesseract can detect all number, alphabets, special symbols

and it can also detect other languages when other language is set in function. When a text is detected that text gets printed in the terminal.

When all 4 fingers are up all paint is removed, this is done by again setting canvas as zero array.

Problems faced:

1) Initially when the program was configured to recognize the drawn text live, it suffered a lot of lag and the accuracy was also compromised. Therefore, the detection mode was added wherein the text was detected only when instructed by the user when he is done writing.

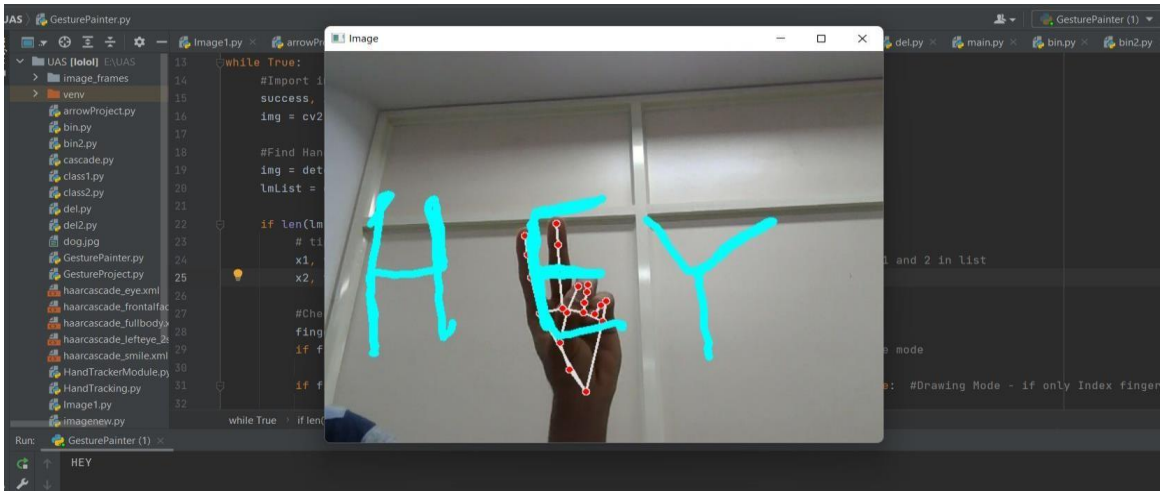
2) An error was obtained when we were trying to perform bitwise AND and bitwise OR operation on image and canvas as the size of the two arrays were not equal (which is a compulsory condition for the operation to be performed). Then the values of the size of the two were adjusted using hit and try to obtain the desired values.

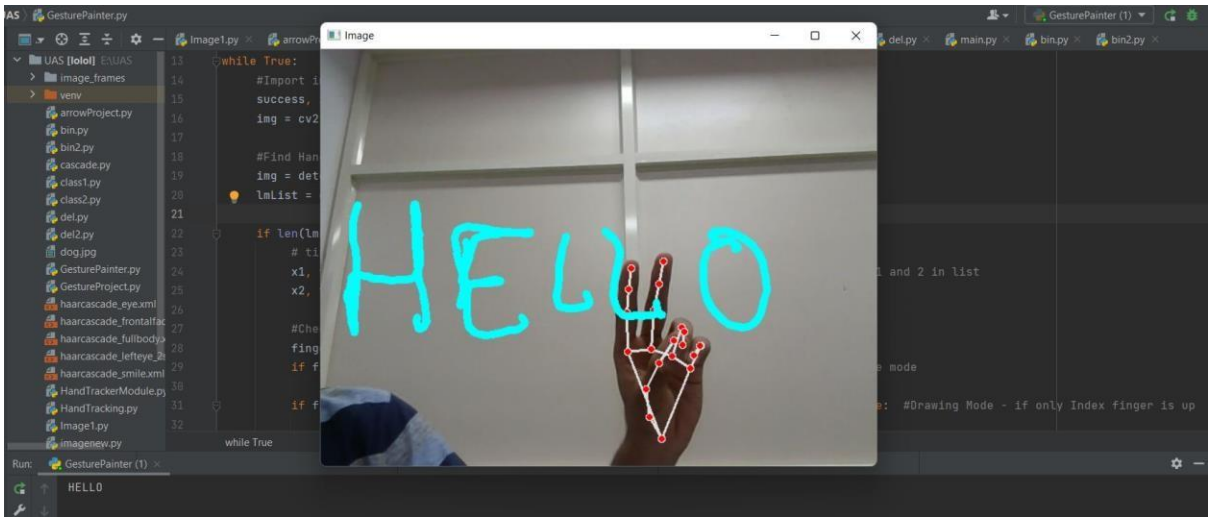
3) Mphands.hands() takes 5 variables in newer version of mediapipe earlier it took 4 variables, so we were getting error when we were only taking 4 variables. (new variable introduced was min_tracking_confidence)

Possible improvements:

1) To make the program able to recognize the text when characters are drawn at different vertical positions.
2) Adding an erasing mode so that the whole canvas doesn't needs to be cleared.

IMAGES





Terminal