Top of Form

Bottom of Form

|  |  |
| --- | --- |
|  | import mediapipe as mp |
|  | import numpy as np |
|  | from flask import Flask, render\_template, request |
|  | import cv2 |
|  | import os |
|  | from keras.models import load\_model |
|  | from werkzeug.utils import secure\_filename |
|  |  |
|  |  |
|  | app = Flask(\_\_name\_\_) |
|  | @app.route("/") |
|  | def home(): |
|  | return render\_template("home.html") |
|  |  |
|  | @app.route("/process", methods=['GET', 'POST']) |
|  | def process(): |
|  | if request.method == 'POST': |
|  | upload\_image = request.files['upload\_image'] |
|  | basepath=os.path.dirname(\_\_file\_\_) |
|  | file\_path=os.path.join(basepath,'static',secure\_filename(upload\_image.filename)) |
|  | upload\_image.save(file\_path) |
|  | model1 = load\_model('gesture.h5') |
|  | mpHands = mp.solutions.hands |
|  | hands = mpHands.Hands(max\_num\_hands=1, min\_detection\_confidence=0.5, min\_tracking\_confidence=0.5) |
|  | mpDraw = mp.solutions.drawing\_utils |
|  | cap = cv2.VideoCapture(0) |
|  | while True: |
|  | \_, frame = cap.read() |
|  |  |
|  | h, w, c = frame.shape |
|  |  |
|  | frame = cv2.flip(frame, 1) |
|  | framergb = cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB) |
|  |  |
|  | result = hands.process(framergb) |
|  | res = '' |
|  |  |
|  | if result.multi\_hand\_landmarks: |
|  | landmarks = [] |
|  | for handslms in result.multi\_hand\_landmarks: |
|  | x\_max = 0 |
|  | y\_max = 0 |
|  | x\_min = w |
|  | y\_min = h |
|  | for lm in handslms.landmark: |
|  | x = int(lm.x \* w) |
|  | y = int(lm.y \* h) |
|  |  |
|  | landmarks.append([x, y]) |
|  | if x > x\_max: |
|  | x\_max = x |
|  | if x < x\_min: |
|  | x\_min = x |
|  | if y > y\_max: |
|  | y\_max = y |
|  | if y < y\_min: |
|  | y\_min = y |
|  | cv2.rectangle(frame, (x\_min - 5, y\_min - 5), (x\_max + 5, y\_max + 5), (0, 255, 0), 2) |
|  | framegray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY) |
|  | hand = framegray[y\_min - 5:y\_max + 5, x\_min - 5:x\_max + 5] |
|  | hand = cv2.resize(hand, (128, 128)) |
|  | hand = hand / 255 |
|  | hand = hand.reshape(128, 128, 1) |
|  | hand = np.expand\_dims(hand, axis=0) |
|  | mpDraw.draw\_landmarks(frame, handslms, mpHands.HAND\_CONNECTIONS) |
|  | prediction = model1.predict(hand) |
|  | res = np.argmax(prediction) |
|  | image1 = cv2.imread(file\_path) |
|  | image1= cv2.resize(image1,(300,300)) |
|  | if res==1: |
|  | resized = cv2.resize(image1, (200, 200)) |
|  | cv2.imshow("Resizing", resized) |
|  | key=cv2.waitKey(3000) |
|  |  |
|  | if (key & 0xFF) == ord("1"): |
|  | cv2.destroyWindow("Resizing") |
|  |  |
|  | elif res==2: |
|  | blurred = cv2.GaussianBlur(image1, (21, 21), 0) |
|  | cv2.imshow("Blurred", blurred) |
|  | key=cv2.waitKey(3000) |
|  | if (key & 0xFF) == ord("3"): |
|  | cv2.destroyWindow("Blurred") |
|  |  |
|  | elif res==3: |
|  | (h, w, d) = image1.shape |
|  | center = (w // 2, h // 2) |
|  | M = cv2.getRotationMatrix2D(center, -45, 1.0) |
|  | rotated = cv2.warpAffine(image1, M, (w, h)) |
|  | cv2.imshow("OpenCV Rotation", rotated) |
|  | key=cv2.waitKey(3000) |
|  | if (key & 0xFF) == ord("2"): |
|  | cv2.destroyWindow("OpenCV Rotation") |
|  |  |
|  |  |
|  | elif res==4: |
|  | cv2.rectangle(image1, (480, 170), (650, 420), (0, 0, 255), 2) |
|  | cv2.imshow("Rectangle", image1) |
|  | cv2.waitKey(0) |
|  | key=cv2.waitKey(3000) |
|  | if (key & 0xFF) == ord("0"): |
|  | cv2.destroyWindow("Rectangle") |
|  |  |
|  |  |
|  | else: |
|  | continue |
|  |  |
|  |  |
|  | cv2.imshow("Output", frame) |
|  |  |
|  | if cv2.waitKey(1) == ord('q'): |
|  | break |
|  |  |
|  | cap.release() |
|  | cv2.destroyAllWindows() |
|  |  |
|  | return render\_template("index.html") |
|  |  |
|  |  |
|  | @app.route("/intro") |
|  | def intro\_page(): |
|  | return render\_template("intro.html") |
|  |  |
|  | @app.route("/index") |
|  | def index\_page(): |
|  | return render\_template("index.html") |