import pickle

import cv2

import mediapipe as mp

import numpy as np

import pygame

pygame.mixer.init()

model\_dict = pickle.load(open('./model.p', 'rb'))

model = model\_dict['model']

cap = cv2.VideoCapture(0)

mp\_hands = mp.solutions.hands

mp\_drawing = mp.solutions.drawing\_utils

mp\_drawing\_styles = mp.solutions.drawing\_styles

hands = mp\_hands.Hands(static\_image\_mode=True, min\_detection\_confidence=0.3)

labels\_dict = {0: 'Bathroom', 1: 'No', 2: 'Hello',3:'Sorry',4:'Stop',5:'What',6:'Help',7:'Where',8:'Correct',9:'Cow',10:'Telephone',11:'Thank You',12:'I Love You',13:'Eat',14:'Yes',15:'Fever'}

# Load audio files

audio\_files = {

    'Bathroom': pygame.mixer.Sound('./audio/bathroom.mp3'),

    'No': pygame.mixer.Sound('./audio/no.mp3'),

    'Hello': pygame.mixer.Sound('./audio/hello.mp3'),

    'Sorry': pygame.mixer.Sound('./audio/sorry.mp3'),

    'Stop': pygame.mixer.Sound('./audio/stop.mp3'),

    'What': pygame.mixer.Sound('./audio/what.mp3'),

    'Help': pygame.mixer.Sound('./audio/help.mp3'),

    'Where': pygame.mixer.Sound('./audio/where.mp3'),

    'Correct': pygame.mixer.Sound('./audio/correct.mp3'),

    'Cow': pygame.mixer.Sound('./audio/cow.mp3'),

    'Telephone': pygame.mixer.Sound('./audio/telephone.mp3'),

    'Thank You': pygame.mixer.Sound('./audio/thankyou.mp3'),

    'I Love You': pygame.mixer.Sound('./audio/iloveyou.mp3'),

    'Eat': pygame.mixer.Sound('./audio/eat.mp3'),

    'Yes': pygame.mixer.Sound('./audio/yes.mp3'),

    'Fever': pygame.mixer.Sound('./audio/fever.mp3'),

}

# Variable to keep track of whether a sound is currently playing

sound\_playing = False

last\_predicted\_character = None

while True:

    data\_aux = []

    x\_ = []

    y\_ = []

    ret, frame = cap.read()

    H, W, \_ = frame.shape

    frame\_rgb = cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB)

    results = hands.process(frame\_rgb)

    if results.multi\_hand\_landmarks:

        for hand\_landmarks in results.multi\_hand\_landmarks:

            mp\_drawing.draw\_landmarks(

                frame,  # image to draw

                hand\_landmarks,  # model output

                mp\_hands.HAND\_CONNECTIONS,  # hand connections

                mp\_drawing\_styles.get\_default\_hand\_landmarks\_style(),

                mp\_drawing\_styles.get\_default\_hand\_connections\_style())

        for hand\_landmarks in results.multi\_hand\_landmarks:

            for i in range(len(hand\_landmarks.landmark)):

                x = hand\_landmarks.landmark[i].x

                y = hand\_landmarks.landmark[i].y

                x\_.append(x)

                y\_.append(y)

            for i in range(len(hand\_landmarks.landmark)):

                x = hand\_landmarks.landmark[i].x

                y = hand\_landmarks.landmark[i].y

                data\_aux.append(x - min(x\_))

                data\_aux.append(y - min(y\_))

        x1 = int(min(x\_) \* W) - 10

        y1 = int(min(y\_) \* H) - 10

        x2 = int(max(x\_) \* W) - 10

        y2 = int(max(y\_) \* H) - 10

        if len(data\_aux) != 42:

            warning\_message = "Warning: Detected " + str(len(data\_aux)) + " features. Expected 42 features."

            cv2.putText(frame, warning\_message, (50, 50), cv2.FONT\_HERSHEY\_SIMPLEX, 1, (0, 0, 255), 2, cv2.LINE\_AA)

        else:

            prediction = model.predict([np.asarray(data\_aux)])

            predicted\_character = labels\_dict[int(prediction[0])]

            if predicted\_character in audio\_files and not sound\_playing and predicted\_character != last\_predicted\_character:

                audio\_files[predicted\_character].play()

                sound\_playing = True

                last\_predicted\_character = predicted\_character

            if sound\_playing and not pygame.mixer.get\_busy():

                sound\_playing = False

            cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 0, 0), 4)

            cv2.putText(frame, predicted\_character, (x1, y1 - 10), cv2.FONT\_HERSHEY\_SIMPLEX, 1.3, (0, 0, 0), 3,

                        cv2.LINE\_AA)

    cv2.imshow('frame', frame)

    cv2.waitKey(1)

cap.release()

cv2.destroyAllWindows()