**Background video remover code**

**1.from flask import Flask, render\_template, Response**

**import cv2**

**import cvzone**

**from cvzone.SelfiSegmentationModule import SelfiSegmentation**

**import os**

The code begins by importing the necessary libraries for the application, including Flask for web server functionality, cv2 for computer vision operations, cv zone for additional computer vision utilities, and the Selfie Segmentation module from c zone for background removal. The os module is also imported to work with file paths and directories.

**2.app = Flask(\_\_name\_\_)**

Here, an instance of the Flask class is created, representing the Flask application. The \_\_name\_\_ variable is passed as the argument, which refers to the current module or package.

**3.segmentor = SelfiSegmentation()**

**fpsReader = cvzone.FPS()**

Instances of the SelfiSegmentation class and FPS class from cvzone are created. segmentor is used for background removal, and fpsReader is used to calculate and display the frames per second (FPS) of the processed video.

**4.listImg = os.listdir("Images")**

**print(listImg)**

**imgList = []**

**for imgPath in listImg:**

**img = cv2.imread(f'Images/{imgPath}')**

**imgList.append(img)**

**print(len(imgList))**

The code retrieves a list of image file names from the "Images" directory using os.listdir(). It then creates an empty list imgList to store the images. The code loops through each image file, reads it using cv2.imread(), and appends it to the imgList. The length of imgList is printed, which represents the number of background images available for segmentation.

**5.indexImg = 0**

A variable indexImg is initialized to keep track of the current index of the background image being used for segmentation.

**6.def gen\_frames():**

**cap = cv2.VideoCapture(0)**

**cap.set(3, 640)**

**cap.set(4, 480)**

**cap.set(cv2.CAP\_PROP\_FPS, 60)**

**while True:**

**success, img = cap.read()**

**if not success:**

**break**

**imgOut = segmentor.removeBG(img, imgList[indexImg], threshold=0.89)**

**imgStacked = cvzone.stackImages([img, imgOut], 2, 1)**

**\_, imgStacked = fpsReader.update(imgStacked, color=(0, 0, 255))**

**ret, buffer = cv2.imencode('.jpg', imgStacked)**

**frame = buffer.tobytes()**

**yield (b'--frame\r\n'**

**b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n')**

**cap.release()**

This is a generator function gen\_frames() that captures video frames. It begins by creating a VideoCapture object cap to access the default camera (index 0). The resolution is set to 640x480, and the FPS is set to 60.

Inside the while loop, it reads frames from the camera using cap.read(). If the frame is successfully read, the code proceeds to remove the background using segmentor.removeBG() from the SelfiSegmentation module. The imgList[indexImg] represents the current background image being used. Adjusting the threshold value allows for different levels of background removal.

The processed frame and the original frame are stacked horizontally using cvzone.stackImages(), and the resulting image is passed to fpsReader.update() to calculate the FPS and display it on the image. The frame is then converted to JPEG format using cv2.imencode(), and the bytes are yielded using the yield statement.

**7.@app.route('/')**

**def index():**

**return render\_template('index.html')**

**@app.route('/video\_feed')**

**def video\_feed():**

**return Response(gen\_frames(), mimetype='multipart/x-mixed-replace; boundary=frame')**

Two routes are defined using Flask decorators. The '/' route renders the index.html template, and the '/video\_feed' route returns the response with the frames generated by gen\_frames(). The response is set with the appropriate mimetype for a continuous video stream.

**8.if \_\_name\_\_ == '\_\_main\_\_':**

**app.run(debug=True)**

Finally, the application is run by calling app.run(). The debug=True argument enables debug mode, which provides useful information in case of errors or issues.

This code sets up a Flask application to perform real-time background removal using the SelfiSegmentation module. It captures frames from the camera, applies background removal, and streams the processed frames to a web page. The background removal is based on a series of background images stored in the "Images" directory.