Real-Time Background Change Using OpenCV and MediaPipe

# Overview

This project demonstrates how to replace the real-time background of a webcam feed with a custom image using OpenCV and MediaPipe. MediaPipe’s Selfie Segmentation model identifies the foreground (human) and background, enabling the background replacement.

# Code Breakdown

## 1. Import Necessary Modules

import cv2  
import mediapipe as mp  
  
- \*\*cv2\*\*: OpenCV module for image processing and video feed handling.  
- \*\*mediapipe\*\*: Provides pre-trained models, including Selfie Segmentation, for various AI tasks.

## 2. Initialize MediaPipe Selfie Segmentation

mp\_selfie\_segmentation = mp.solutions.selfie\_segmentation  
segmenter = mp\_selfie\_segmentation.SelfieSegmentation(model\_selection=1)  
  
- \*\*`mp\_selfie\_segmentation`\*\*: Access the MediaPipe Selfie Segmentation module.  
- \*\*`SelfieSegmentation(model\_selection=1)`\*\*: Creates the segmentation object.  
 - `model\_selection=0`: Optimized for selfies.  
 - `model\_selection=1`: General use, separates background and foreground effectively.

## 3. Access Webcam Feed

cap = cv2.VideoCapture(0)  
  
- Captures video feed from the default webcam (camera index `0`).

## 4. Load Background Image

bg\_img = cv2.imread("bg.jpg")  
if bg\_img is None:  
 print("Error: Unable to Access Background image")  
 exit()  
  
- \*\*`cv2.imread()`\*\*: Loads the background image.  
- If the image isn’t found, an error message is printed, and the program exits.

## 5. Process Webcam Feed

while cap.isOpened():  
 ret, frame = cap.read()  
 frame = cv2.flip(frame, 1)  
 if not ret:  
 print("Unable to access webcam :(")  
 break  
  
- \*\*`cap.isOpened()`\*\*: Checks if the webcam feed is available.  
- \*\*`cap.read()`\*\*: Captures each frame from the webcam.  
- \*\*`cv2.flip(frame, 1)`\*\*: Flips the frame horizontally for a mirror effect.

## 6. Convert to RGB

frame\_rgb = cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB)  
  
- OpenCV uses the BGR color format by default. MediaPipe requires RGB, so the frame is converted.

## 7. Process Segmentation

result = segmenter.process(frame\_rgb)  
if result.segmentation\_mask is not None:  
 mask = result.segmentation\_mask  
  
- \*\*`segmenter.process(frame\_rgb)`\*\*: Processes the frame to detect the foreground and background, creating a segmentation mask.  
- \*\*`segmentation\_mask`\*\*: MediaPipe’s output. It is a grayscale mask where pixel values indicate the likelihood of being foreground (1) or background (0).

## 8. Resize Background Image

bg\_resized = cv2.resize(bg\_img, (frame.shape[1], frame.shape[0]))  
  
- Resizes the background image to match the dimensions of the webcam feed.

## 9. Threshold the Mask

mask = (mask > 0.5).astype('uint8')  
  
- \*\*`mask > 0.5`\*\*: Converts the segmentation mask into a binary mask. Pixels greater than 0.5 are set to 1 (foreground), and others are set to 0 (background).  
- \*\*`astype('uint8')`\*\*: Converts the binary mask to 8-bit format (values between 0 and 255) for OpenCV compatibility.

## 10. Create Inverted Mask

mask\_inv = cv2.bitwise\_not(mask \* 255)  
  
- \*\*`mask \* 255`\*\*: Scales the binary mask values (0 or 1) to 0 or 255.  
- \*\*`cv2.bitwise\_not()`\*\*: Inverts the mask. Foreground becomes black, and background becomes white.

## 11. Extract Foreground (Person)

person = cv2.bitwise\_and(frame, frame, mask=mask)  
  
- Combines the frame and the mask using a bitwise AND operation, keeping only the foreground (person).

## 12. Extract Background

background\_convert = cv2.bitwise\_and(bg\_resized, bg\_resized, mask=mask\_inv)  
  
- Combines the resized background image and the inverted mask using a bitwise AND operation, keeping only the background.

## 13. Combine Foreground and Background

converted = cv2.add(person, background\_convert)  
  
- Combines the extracted foreground (person) and background to create the final output.

## 14. Display the Output

cv2.imshow('Custom Background media', converted)  
  
- Displays the final frame with the custom background applied.

## 15. Handle Exit Condition

if cv2.waitKey(1) == 27:  
 cv2.destroyAllWindows()  
 cap.release()  
 break  
  
- \*\*`cv2.waitKey(1)`\*\*: Waits for a key press. `27` is the ASCII code for the `Esc` key.  
- \*\*`cv2.destroyAllWindows()`\*\*: Closes all OpenCV windows.  
- \*\*`cap.release()`\*\*: Releases the webcam.

# Final Output

The real-time video feed replaces the original background with the custom image (`bg.jpg`), keeping only the person visible.  
