import cv2

# Load the two images

img1 = cv2.imread("image1.jpg")

img2 = cv2.imread("image2.jpg")

# Detect faces in the images

face\_cascade = cv2.CascadeClassifier("haarcascade\_frontalface\_default.xml")

face1 = face\_cascade.detectMultiScale(img1, 1.3, 5)

face2 = face\_cascade.detectMultiScale(img2, 1.3, 5)

# Crop the face from the images

for (x,y,w,h) in face1:

img1\_face = img1[y:y+h, x:x+w]

for (x,y,w,h) in face2:

img2\_face = img2[y:y+h, x:x+w]

# Create a mask for the face

img1\_face = cv2.cvtColor(img1\_face, cv2.COLOR\_BGR2GRAY)

ret, mask = cv2.threshold(img1\_face, 220, 255, cv2.THRESH\_BINARY\_INV)

# Create an inverse mask

mask\_inv = cv2.bitwise\_not(mask)

# Use the mask to extract the face from the background

img1\_bg = cv2.bitwise\_and(img1, img1, mask=mask\_inv)

img2\_fg = cv2.bitwise\_and(img2\_face, img2\_face, mask=mask)

# Put the extracted face on top of the background

dst = cv2.add(img1\_bg, img2\_fg)

# Show the final image

cv2.imshow("Face Swap", dst)

cv2.waitKey(0)

cv2.destroyAllWindows()

This code does the following:

1. loads two images using the imread function
2. Detects faces in the images using a pre-trained classifier
3. Crop the face from the images
4. Create a mask for the face
5. Create an inverse mask
6. Use the mask to extract the face from the background
7. Put the extracted face on top of the background
8. Show the final image

This is a basic example and can be improved by different techniques like:

* Using Dlib or MTCNN for face detection
* Using facial landmarks to align the faces
* Using seamless cloning for blending the swapped face with the background.

I hope this helps! Let me know if you have any other questions.

Top of Form