A project Report on

Face Swap

using machine learning

CS550: Machine Learning

Instructor:Dr. SK Subidh Ali
TAs: Indrakumar Mhaski, Sudev Kumar Padhi

Group name:
Conquerors

Group members:
Ashish Kumar Suraj(11840230),
Vikash Vitthore(11841230)

1. Abstract

Swapping faces means transferring a face from a source photo onto a face appearing in a target photo, attempting to generate realistic, unedited looking results. This project implements an image-to-image face swapping but the target would be real time webcam where faces would be replaced by the source image. In this project, we present a complete system for automatic face replacement in the webcam. Face swapping algorithms can be roughly divided into three categories: replacement-based, model-based, and learning-based. The Used algorithm is composed of three steps: face alignment, warping and replacement. The accuracy and robustness of the algorithm are enhanced by introducing some learning-based modules like facial landmark detection and face parsing.

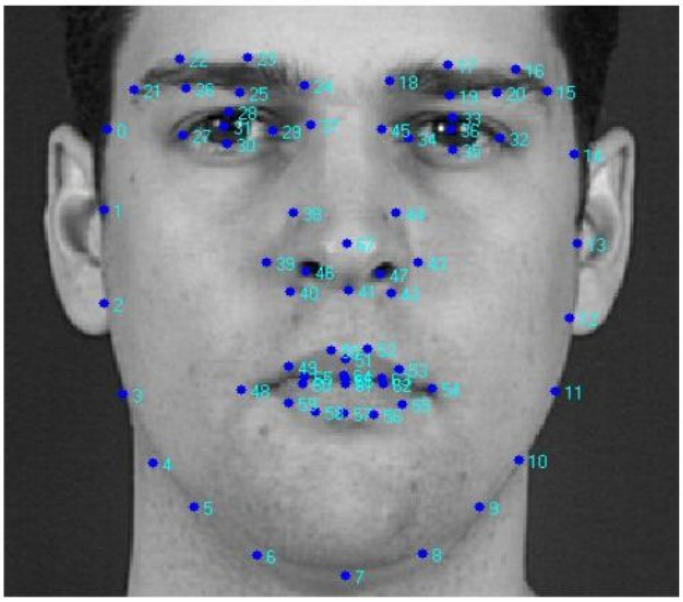
2. Introduction

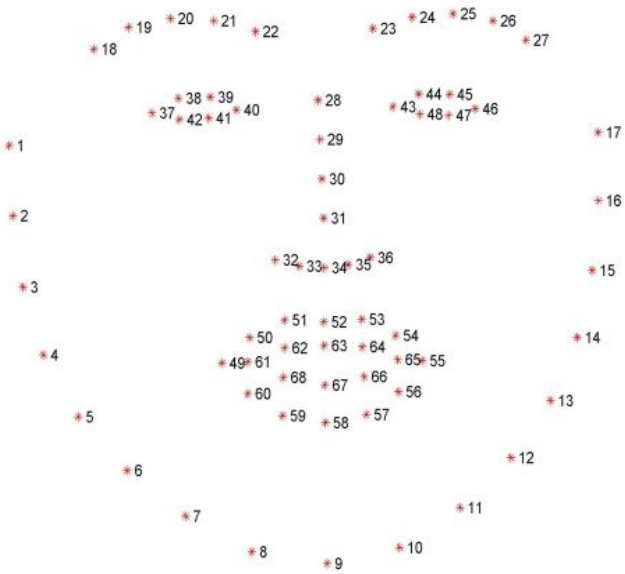
Face swapping has been a fascinating problem for image processing researchers during the last decade because of many important applications such as video face swapping, digital image archiving, etc. Advances in digital photography have made it possible to capture large collections of high-resolution images and share them on the internet. Talking about the importance of the program, it is very useful in many types of fiend such as Google Street view. In that you are able to go through all the streets and view the view. Inside that you will also be getting people who were there. Now because of the privacy their face should be swapped with black color or whatever. Here comes the use of this thing face swap. Although this used to swap face to face, but we can detect the face and later on swap it with anything if we have the algorithm. Doing it manually is not so easy, since the number of photos are increasing rapidly, so this is as important as other technologies.

Automatic face replacement has other compelling applications as well. For example, people commonly have large personal collections of photos on their computers. These collections often contain many photos of the same person(s) taken with different expressions, and under various poses and lighting conditions. One can use such collections to create novel images by replacing faces in one image with more appealing faces of the same person from other images. For group shots, the “burst” mode available in most cameras can be used to take several images at a time. With an automatic face replacement approach, one could create a single composite image with, for example, everyone smiling and with both eyes open.

3. Problem Definition

Recognizing 68 points on the face which will contain the boundary of the face and as well as eye, nose and mouth, then swapping the triangles made with the points with the source image.

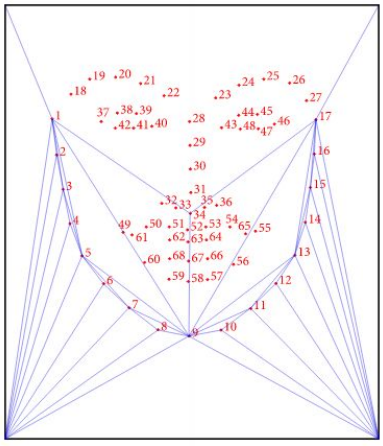


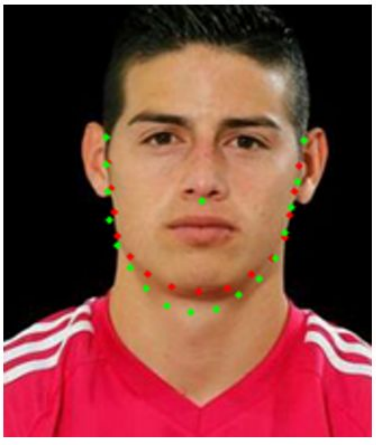


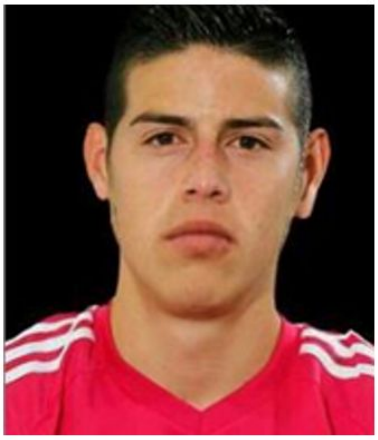
4. Objective

As mentioned above, we would be using the FLD algorithm to solve the problem of swapping the face. There are different steps being used for getting the points on the face and doing later stuff.

a. First it would be taking an image as src_image.
b. We detect 68 landmarks point on face using the dlib library.
c. The points would be containing boundaries and special things on face.
d. Then using those 68 points, we create triangle containing those points on both of the faces.
e. We could be getting a lot of triangles from those points but will try to have as minimum as possible.
f. When we are having the proper triangles of the required place on face, we swap those triangles with the second face.








5. ## Technology Used

For this project we use dlib. It is a modern C++ toolkit containing machine learning algorithms and tools for creating complex software in C++ to solve real world problems. It is also available in python.
Second is OpenCV, this also a deep learning and computer vision technology. With the help of this we crop our images and replace them with other source files.

6. ## Problems Faced

Major problems for this project are detecting face on both source and target image. After that when we create a triangle, then replacing the triangle with the corresponding triangle is challenging because the size or area of both triangles are different. For this we use OpenCV coloring and with the help of we cover the remaining area of the target triangle. Also, we are using dlib library, this library is only fetching the points from the face, We did find methods for recognizing these points. So the problem was to recognize these points from model perspective because all this was done by dlib.

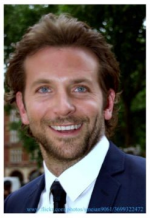
7. ## Datasets Used

For the data-set we use shape_predictor_68_face_landmarks, the data-set is used as 68 points face detection dataset. With the help of dlib, we predict 68 points on both of the faces.

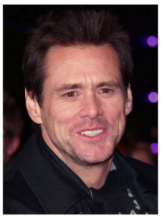
8. ## Working Idea

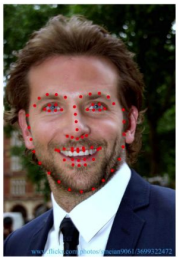
Here are some reference of the working idea behind the project.

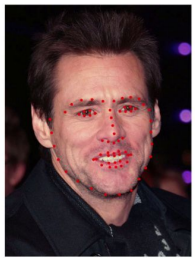
Source Image

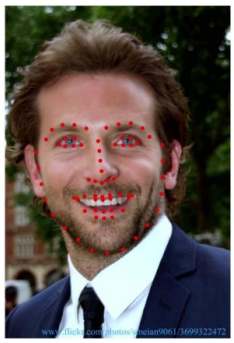


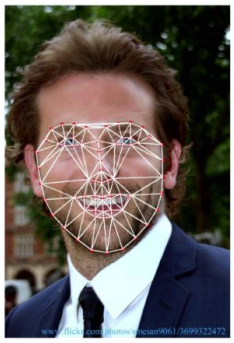
Destination Image

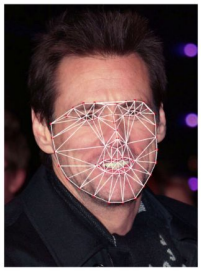


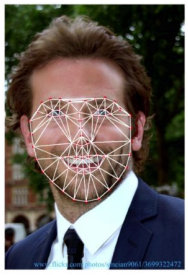












9. Implementation

Since we are using dlib library, we are just giving it the input file as the picture and the file where we have to swap the face. In this case that is our real time webcam.

Using python script image is being processed with the library, and it is giving those 68 points. From those points we are able to create triangles on both faces and then only triangles are being swapped.

10. Result And Performance

The Result getting from the model is pretty good, like we are able to recognize that the face is being swapped and boundary is also visible, but it is doing it in real time with webcam. We were able to swap the face by giving proper inputs. The thing is that the face should be somewhat similar to the face with which we are going to swap then only it will give its best performance.

11. Conclusion And Further Work

In this paper we presented the technique to swap the face in real time. This we know that it would be much harder to do in real time using the model, but instead we used dlib library, and it handled everything. After getting point swapping part is of python file, but before that, work was being handled by the library itself.

Furthermore, we will be trying to get those points by training the model and then giving those points to our python program, we should be able to handle the swap. It is not necessary to do this in real time so most probably we shall try to do it with image to image.

12. References

[1] Research Article | Open Access
https://www.hindawi.com/journals/mpe/2019/8902701/

[2] dlib C++ Library **http://dlib.net/**

[3] OpenCV - OpenCV **https://opencv.org**

[4] deepfakes_faceswap https://github.com/deepfakes/faceswap

[5] FSGAN: Subject Agnostic Face Swapping and Reenactment

[6] Reference to download shape_predictor_68_face_landmarks
http://dlib.net/files/shape_predictor_68_face_landmarks.dat.bz2

[7] Future reference https://github.com/YuvalNirkin/fsgan