Face Swap Project Report

Group name: Conquerors

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

Problem Definition:

The goal of this project is to detect and locate a human face in a color image and swap your real time camera's face with that detected face.

Face swap is the technology of the automatic fusion of two or more different faces into one face. It can be used in fields of video synthesis, privacy protection, picture enhancement, and entertainment applications. We shall be proposing an image-based face swapping algorithm which can be used to replace the face in the reference image with the same facial shape and features as the input face.

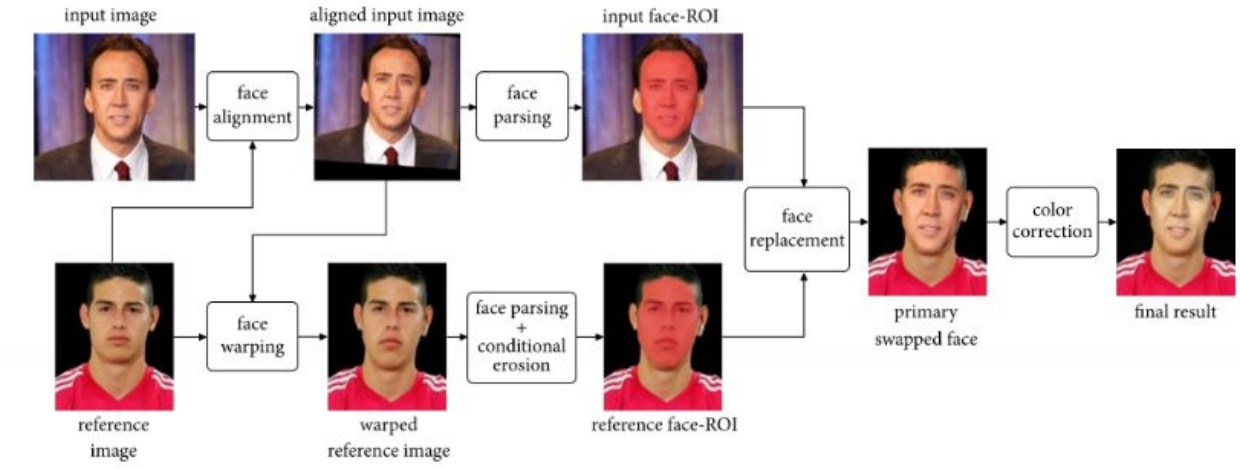
Scope:

The Scope is to generate a good quality real time image from the source image using ML via python programming language.

Models and Methodology:

Building such a project from scratch using the Python programming language would help to achieve a better understanding of the field as well as its advantages and disadvantages compared to other face swapping methods. After some research, the decision to do face swap using an FLD Algorithm and face recognition using 68 facial landmarks was finalized having the project's circumstances in the mind.

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.



These sites gave a lot of helpful information understanding about the project:

- [Face Swapping: Realistic Image Synthesis Based on Facial Landmarks Alignment (hindawi.com)]
- https://pysource.com/2019/05/28/face-swapping-explained-in-8-steps-opencv-with-python

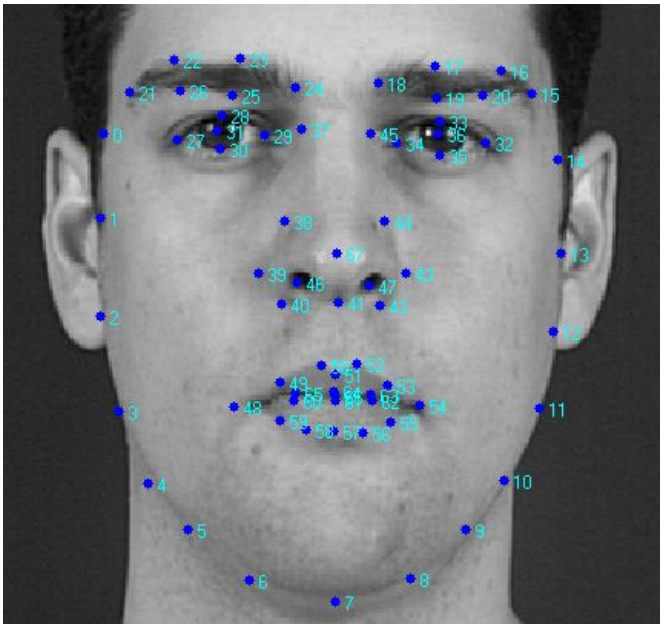
The purpose of this project is to create a program which takes an input image and swaps it with a real time image(webcam)…

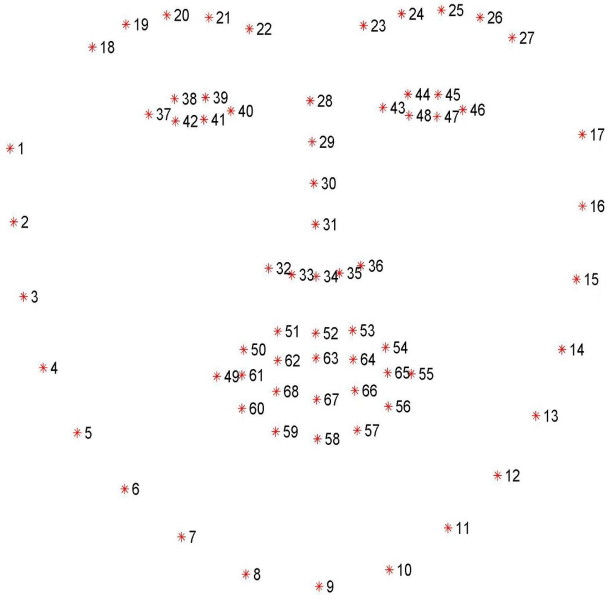
Data collection:

Basically, we are using python predefined library Dlib. In this library we use a dataset of faces. And this method also provides a function for prediction of 68 landmark points. This library works on color changing effects. At the end of the project we try to make our own 68 point detector model. As of now we are using this python function.

Approach:

After detecting all these 68 points on both faces. We found the center of the face with the help of points 22, 23, 49 and 55(see in example image). We can see in our example image our all points are divided into 5 parts eye1, eye2, nose, lip and face boundary points.
Now we convert our detected face into triangles, for this we use a different approach for different landmark points. Like points {1, 2, ..., 17, 34} are using one neighbor and one from the other part out of 4 for making triangles. Similarly, we generate other triangles. And at the end of these, our face is covered by all triangles.





Note:- source image can be an image or real time image

This above process we do with our both files. Note our landmark points are almost similar positions on both images, meaning if point number k is left of the right eye in the source image then similar point number k will be left of the right eye in our destination image. So if a triangle(a1, a2, a2) covers a part in the source image, then a similar triangle also covers almost a similar part of the destination image, the size can be different, for this issue we can use a different approach that is called fitting. So we replace every source triangle with the corresponding destination triangle, and end of this, we swap our total source swap to destination source.

https://youtu.be/MrRGVOhARYY

Implementations done so far:

Till_68_face_landmark_points

```python
#shape_predictor_68_face_landmarks
import cv2
import numpy as np
import dlib

frontal_face_detector = dlib.get_frontal_face_detector()
frontal_face_predictor = dlib.shape_predictor("data/shape_predictor_68_face_landmarks.dat")

webcam_video_stream = cv2.VideoCapture(0)


while (webcam_video_stream.isOpened):

    ret, current_frame = webcam_video_stream.read()
    destination_image = current_frame
    if destination_image is not None:
        destination_image_grayscale = cv2.cvtColor(destination_image, cv2.COLOR_BGR2GRAY)

    if(len(destination_faces)==1):

        for destination_face in destination_faces:

            destination_face_landmarks = frontal_face_predictor(destination_image_grayscale, destination_face)
            destination_face_landmark_points = []


            for landmark_no in range(0,68):
                x_point = destination_face_landmarks.part(landmark_no).x
                y_point = destination_face_landmarks.part(landmark_no).y
                destination_face_landmark_points.append((x_point, y_point))
                print(x_point, y_point)
```