CIS581: Computer Vision & Computational Photography
Face Swapping Final Project Presentation

Deniz Beser, Mia Chiquier, John Wallison

Introduction

This final project aimed to build an algorithm for face swapping. Given two videos, the face swapping algorithm automatically swap faces between the two videos, by detecting faces, and replacing the target faces in the target video with source faces in the source video. The challenging aspect of this problem is keeping the video natural, such as maintaining the emotions of the target face after the transformation. Our implementation of face swapping employed various techniques we learned and used in the class, including CNNs, image morphing, and gradient domain blending.

Methodology

Our method utilizes the Python face recognition and OpenCV libraries to implement the face swapping algorithm. At the core of our methodology is the assumption that emotions are primarily dependent of facial gestures determined by the organization and movements of dominant facial components; by primarily analyzing the positioning and movement of eye, lips, chin, nose, and eyebrows, we can learn the emotion of a face. Consider any emoji; only a few of these facial components often suffice to express an emotion such as a smiley face.

In order to learn the emotion of a face, apply it to a source face, and swap it into the location of the target face, we use the following procedure. We first extract a source face (the face to be inserted in the target video). Then, for each frame in the target video, we analyze the facial gestures of the target image. We form a mapping between the source face to the target face in that frame by applying image warping. Then, we smoothly embed the morphed source face into the target video by applying gradient domain blending.

To extract emotions in the target frame, as well as to analyze the characteristic of the source image, we utilize the Python face recognition library, which allows us to get the locations of faces in a video, as well as a list of facial markers. Then, we use our code from Project 2 (Image Warping) to morph the source image, with its gestures, in the to target image in each frame. Finally, we use the OpenCV library to do gradient domain blending (equivalent to our code from Project 3 for seamless cloning) to produce a smooth embedding of the source face into the target video. Overall, applying this procedure allows us to recreate the target video with the source face. The process is applied in the same fashion to produce multiple face swaps when needed.

Video: example output

Future work & References

Although our simple effectively pipeline performs face swapping, there are numerous other ways that one can approach the problem and refine the results. Incorporating other methods of image transformation (instead of morphing) such as a secondary level of neural architecture or homographic transformations is one way to potentially improve the model.

- Implementation (github): https://github.com/jandwally/nn-faceswap
- Face recognition API: https://github.com/ageitgey/face_recognition
- OpenCV: https://www.learnopencv.com/seamless-cloning-using-opencv-python-cpp/