Face Swapping: An implementation with Dlib and OpenCV

Hongrui Zheng, Tarmily Wen, Shraddha Jain

I. INTRODUCTION

The goal of this project was to implement face swapping between two videos. We went with a route that preserves the original facial expression in the background video, only swapping in the facial features of the face that we're cutting out of the other video.

II. METHODOLOGY

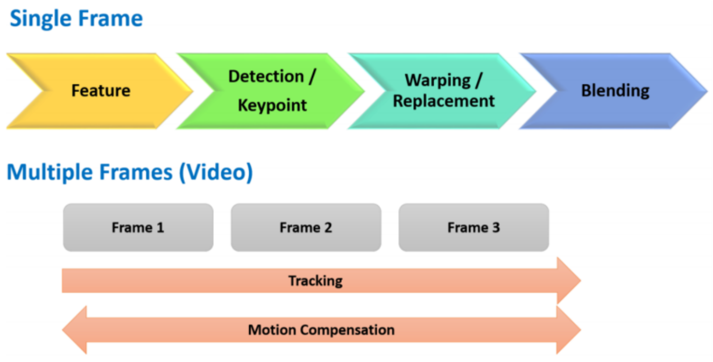

Fig. 1. The pipeline of face swapping

The pipeline of the face swapping process is shown in Figure 1. We used several methods that Dlib provided. The difference between our implementation and most of the other implementations shown in class is that we only used one frame of the video that contains the face that we want to swap on. The same face pixels are used to be morphed and translated into different locations and shapes according to the frame of the video that contains the body. This strategy yielded unexpected results that will be shown in a later section. Also, our code was designed to take in command line arguments as the input for the pipeline. In the beginning of the pipeline, our code shows a frame with the landmarks detected and prompt the user to determine if a frame has good detection, and if that frame is to be used for face extraction. An example of the prompt is shown in Figure 2.

A. Face Detection

For face detection, both the CNN face detector and detector utilizing Histogram of Oriented Gradients(HOG) feature combined with a linear classifier. In our experiments with the videos, the HOG-based detector did just as well as the CNN-based detector and is much faster when running on a machine without a GPU. We decided to use the HOG-based detector.

B. Facial Landmark Detection

For key point detection, we used the shape predictor provided by Dlib. The shape predictor is an implementation of [2]. The predictor was trained on iBUG 300-W face landmark dataset. The trained model is loaded prior to starting the pipeline. The facial landmarks are shown in

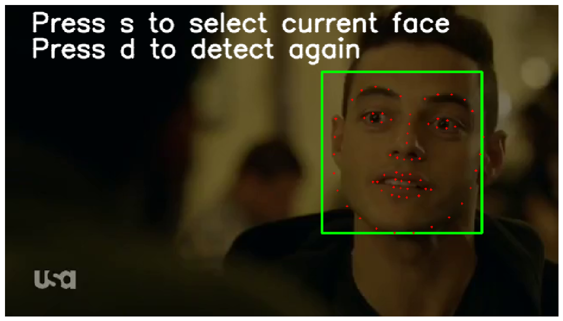

Fig. 2. The facial landmarks

Figure 2. The red points are the predicted landmarks. The predictor always return 68 points that are in the same order in every detection. This came in handy when we needed correspondence between the points to have correct morphing of the face cut out.

C. Feature point tracking

Currently, the facial landmarks are detected in every frame, thus there are no implementation of point tracking. Future improvements are discussed in a section later.

D. Warping

Figure 3 shows a frame in the video that we're swapping the face on to. In this frame, the same 68 points are also detected(not shown in figure) and those facial landmarks are used as the target coordinates for morphing.

Figure 4 shows the warped face. As shown in the figure, the face is warped to the target shape and position, so transformation/translation is not necessary in this case.

Figure 5 shows the face cutout. The cut is determined by the 68 facial landmarks. We first determine the convex hull for the whole face using all the feature points. Then for the eyes and the mouth, only the landmark points around the eye and inside the lips were used to create three convex hulls. The cutout is created by discarding all pixels outside the face convex hull and inside the eye convex hulls and the mouth convex hull.

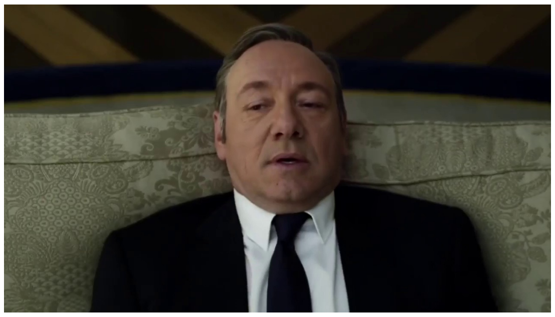
Fig. 3. The body frame

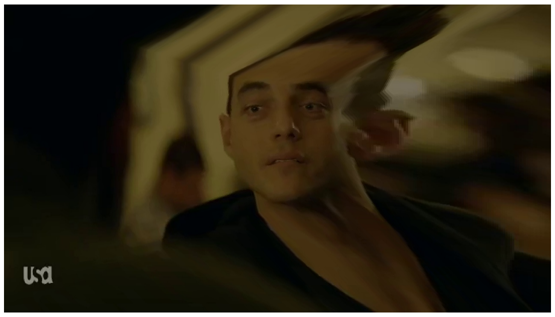
Fig. 4. The warped face

E. Blending

seamlessClone from OpenCV, which is an implementation of [3], and a simple alpha blend is used to blend the face cutout into the body frame. The alpha blend is performed before the gradient blend. A mask is created from the cutout. The alpha blend is performed between the body frame and the cutout. A method is also used to calculate the center of the bounding box around the facial landmarks in the body frame. Then, the resulting frame is used as the source, the mask created is used as the mask, the body frame is used as the target, and the calculated position is used as the clone position in the *seamlessClone*. An example of a blended frame is shown in Figure 6.

III. RESULTS

We've included some video results along with the submission. Our code works surprisingly well on some cases, mediocre on some other cases, and didn't work

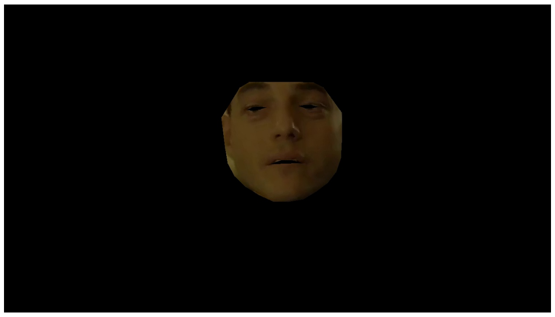
Fig. 5. The face cutout

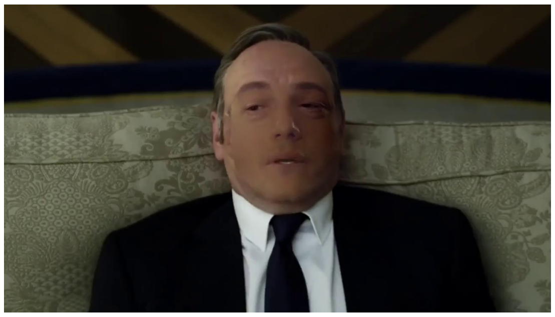
Fig. 6. A blended frame

on extreme cases. For example, in the videos involving either Mr. Robot or Frank Underwood, the code works well because the position of the faces are not drastically changed, and their facial expressions are relatively stable.
The videos involving the Luciano Rosso videos also worked surprisingly well, considering his dramatical facial expressions. The morphing maps the face perfectly to the target area. The facial features with the cutout allows the resulting video to preserve the original facial expressions of the body frame, even in the extreme case of Luciano Rosso. The only case where our method failed was when the facial landmark predictions are off. One solution to this issue is to use Optical Flow to track the landmark points.
Our code doesn't work so well with the videos of Jon Snow, Wolf of Wall Street, and the Joker. The resolution of the Jon Snow video is too low to extract a usable face. Although our code worked with multiple faces, the Leonardo video has obscure faces that are too hard for the detector to pick up. In the Joker video, the face was initially too dark for the detector to pick up. Thus it's almost impossible for our code to put a new face on the Joker video. But it worked when we're extracting the face from this video, and putting it on to other videos, since our code, by design, prompts the user to select a good detection for the face.

IV. FUTURE WORK

A. Optical Flow

Currently, our code doesn't implement point tracking since it might raise a lot of issues and we were unable to resolve all these issues in the limited time we have. One issue is that points will be discarded by Optical Flow if the quality diminishes. Our code needs all 68 points for morphing and cutting out the face. One easy solution is to detect the facial landmarks at the frame where the number of landmarks tracked drops below 68. Another issue is Optical flow may increase the points that overlaps each other, which raises singularity matrices when morphing with Barycentric coordinates. Our code currently deals with this issue by skipping frames. The resulting video might seem unnatural if the number of frames dropped are increased.

B. Live swapping

We have considered swapping the face in a video from a web cam. Our code currently runs decently fast on a machine with a GPU.(Dlib supports GPU accelerating) The speed depends on the resolution of the body frame, since the bottleneck of our code is in morphing and cutout. We have vectorized our code as much as possible, and now it runs at around 1 second per frame if the body video is the Mr. Robot video, and about 4 seconds per frame if the body frame is the Frank Underwood video.

C. 3D mapping

During our research, we've seen methods that utilizes a 3d model in Unity to map the textures on a video. In the future, this method could be used to create better mapping.

REFERENCES

[1] D. E. King, "Max Margin Object Detection", arXiv:1502.00046 [cs.CV]

[2] V. Kazemi, J. Sullivan, "One Millisecond Face Alignment with an Ensemble of Regression Tree", CVPR 2014

[3] P. Perez, M. Gangnet, A. Blake, "Poisson image editing", ACM Transactions on Graphics (SIGGRAPH), 2003