Face Detection and Replacement

Objective

Achieve face detection, feature extraction, face wrapping, face blending, replacement, and finally generate new video with replaced faces.

Algorithm

- Video frames extraction.
- Face detection, feature points extraction (Face ++).
- Face morphing (TPS).
- Face blending (Gradient Domain Blending)
- Face replacement.
- Video synthesis.

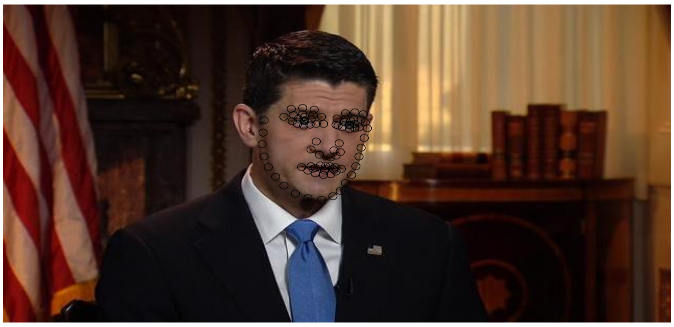
Implementation / Visualization

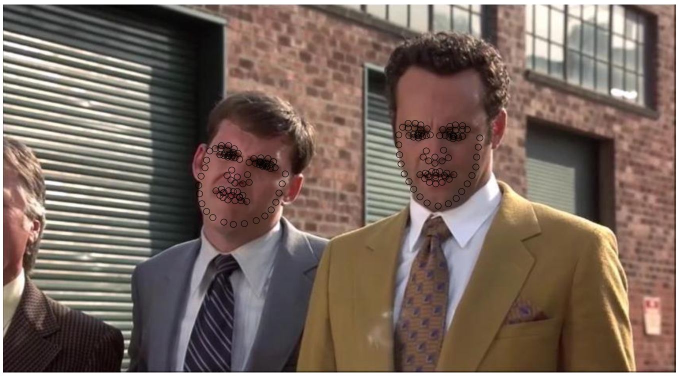
1. Function of Face ++

We utilize the 3rd library **Face ++** mainly on face detection, feature extraction and mask construction.

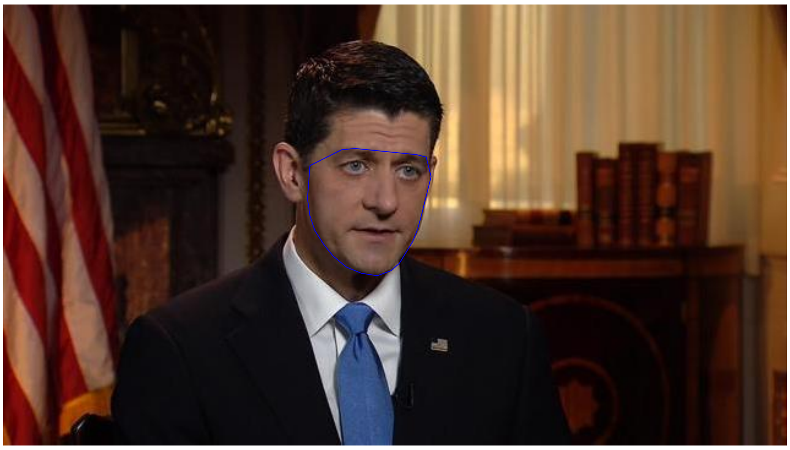
Face ++ will detect faces in image automatically, and provides at most 83 feature points as well. Below are two examples.

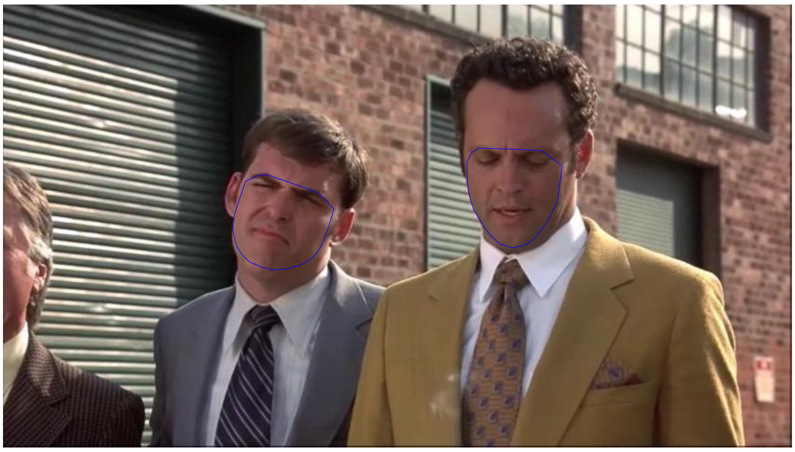
You can see all feature points locate within the face region, what's more, all 83 points are in clockwise order and can get its corresponding points easily.





Then we modifies part of code in face ++ and make it able to generate a logical mask that show value 1 in face region and 0 in background.
The visual results show below. (The blue contour)





Based on these magic mask, we can locate the region that are used for wrapping, blending and replacing easily, and these masks play crucial roles in our whole program.

2. Details of Implementation

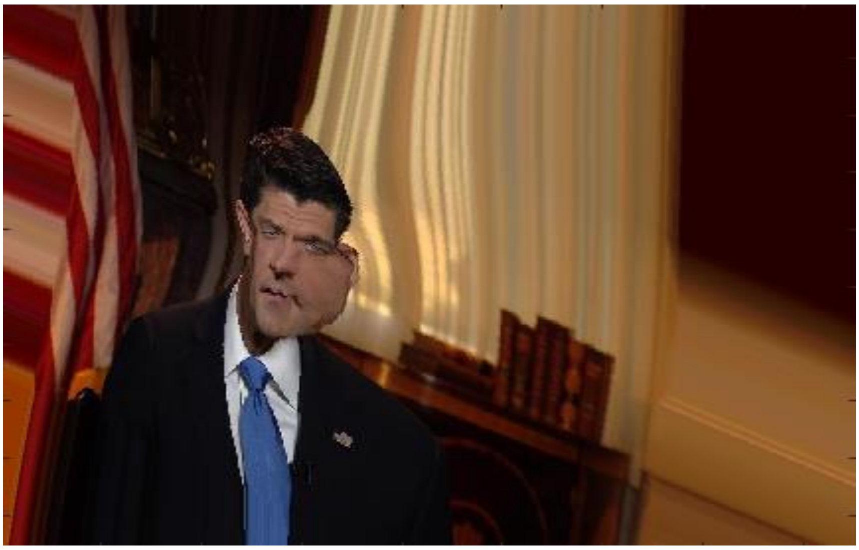
Below is a simple example that shows the principle procedure of our program.

Here is our **source image** that used for replacing:

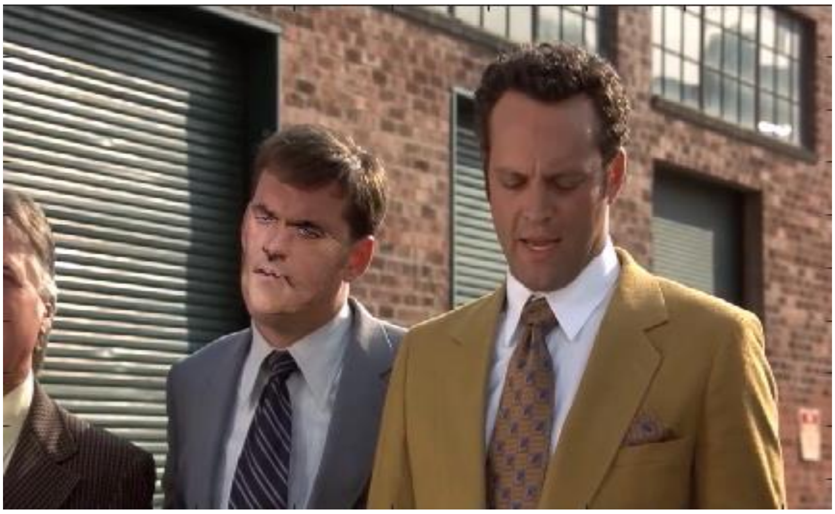
Here is one of frame that extracted from test video and we treat it as **target image.**

After implementation of Face ++, we detect one face in source image and two faces in target image, and here is what we implement.

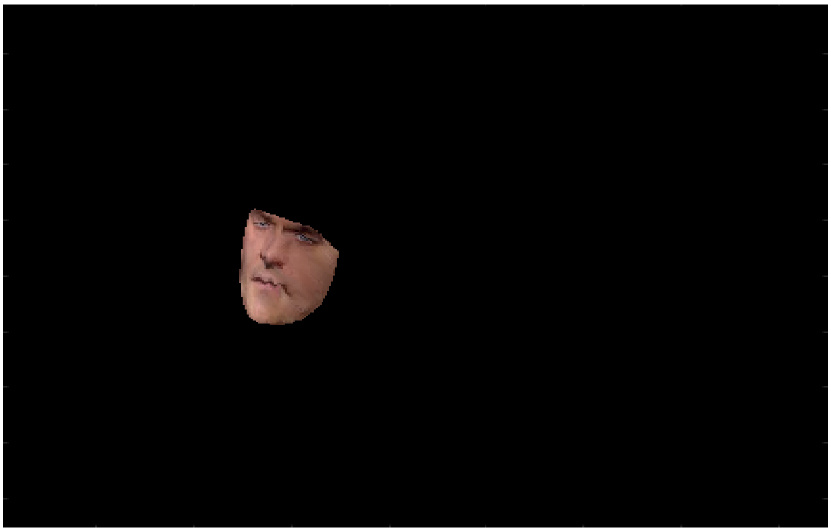
According to feature points in source and target image, we implement **face wrapping** one by one, that is, although we detect two faces in source image, we wrap one of them first, we set it as face 1.



Then take this wrapped image as our new source image, implement **gradient domain blending** for source and target image. So now we can get a new image with only one replaced face.

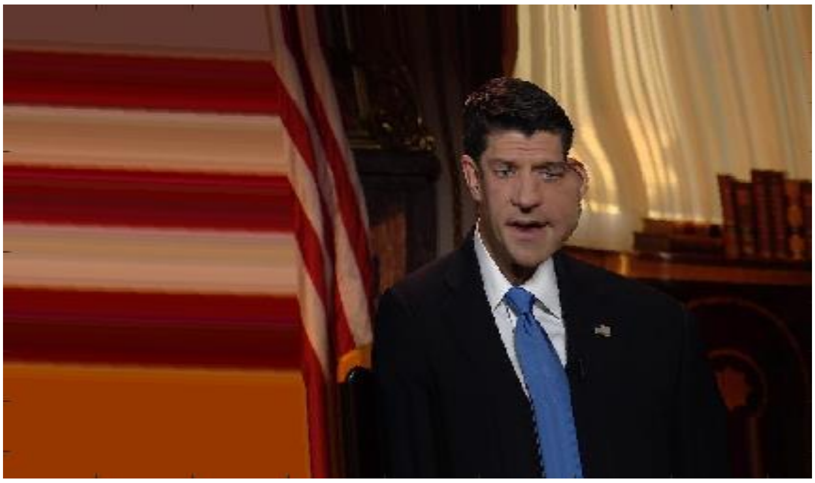


Then we **crop the region that only contains new face out**, via our logical mask that generated from Face ++ and store it.

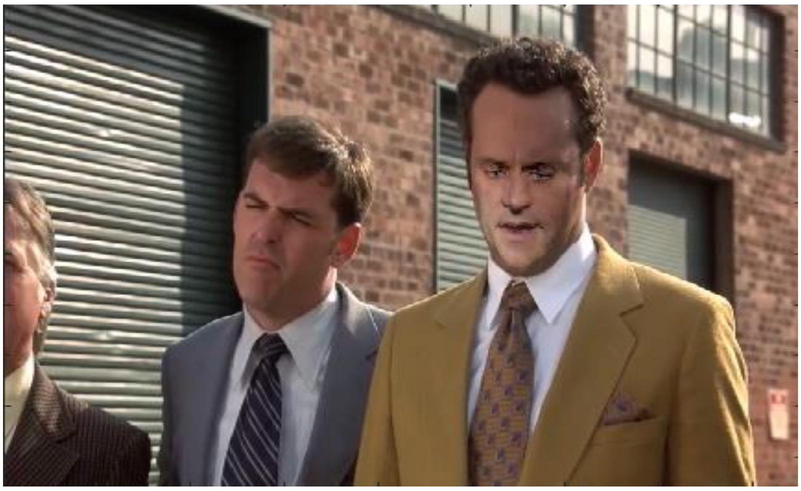


Similarly, we implement same pipeline for second original face and store the replaced one as well.

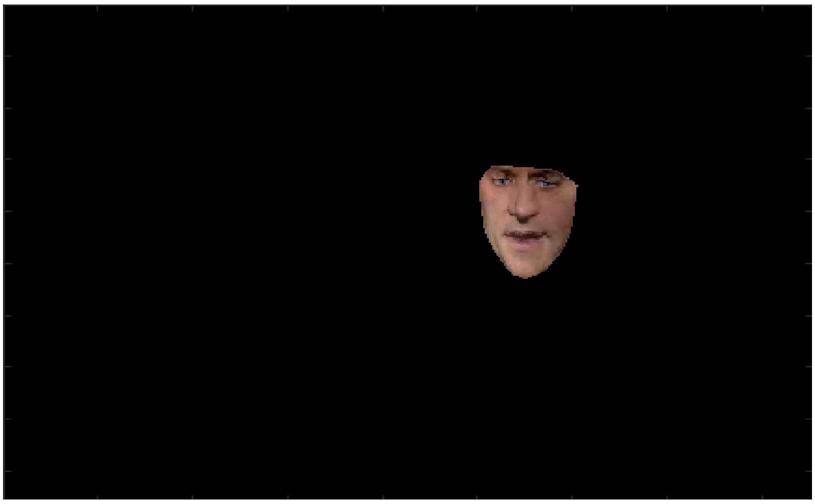
Wrapped source image



Blended image

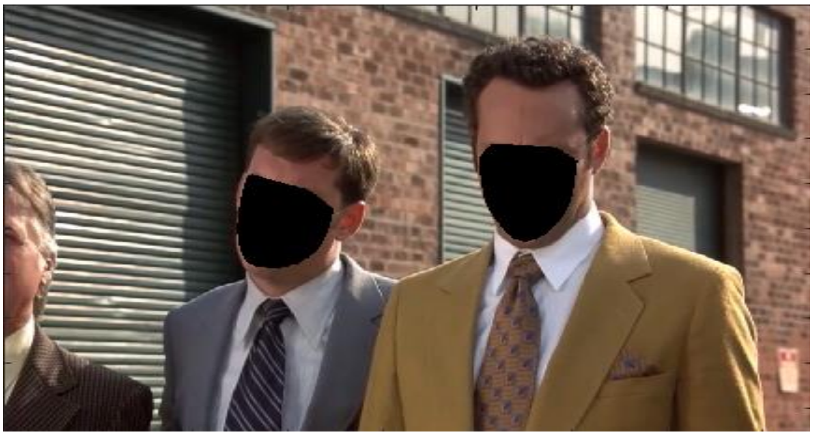


Crop face region

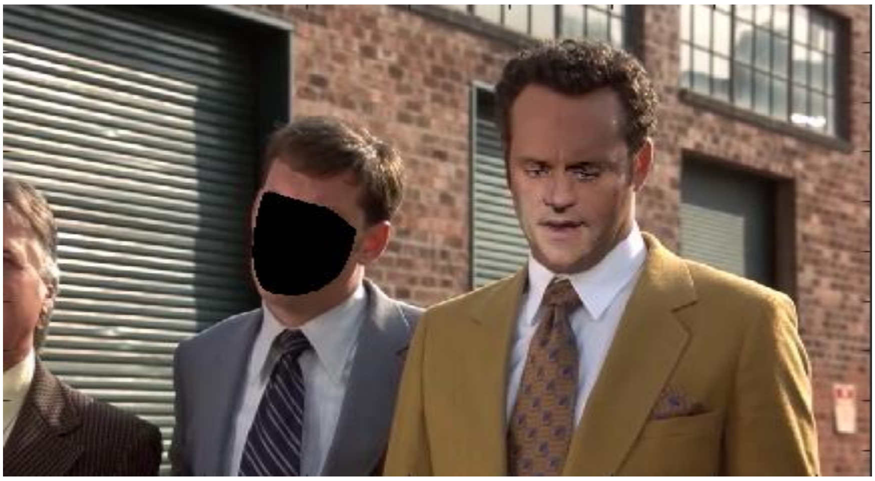


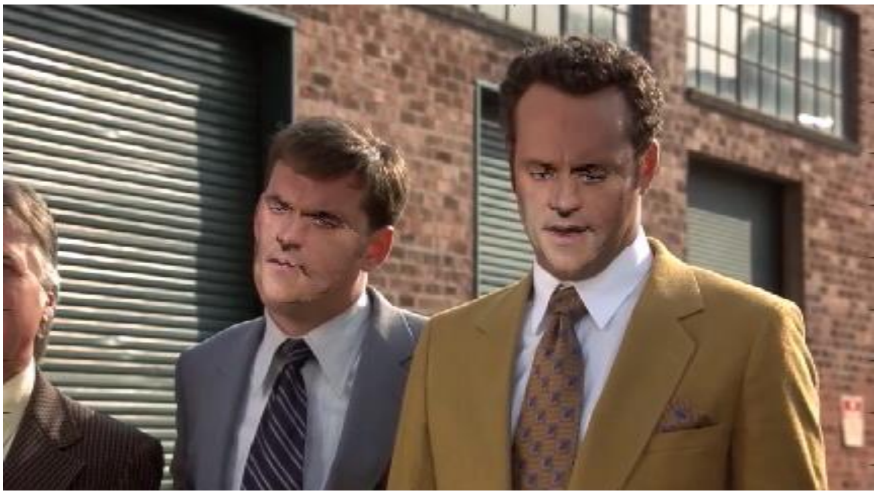
Now dealing with the background region in source image.

Since we just want to replace faces region, therefore we reverse the mask so that background region becomes 1 while face region becomes 0. Then we filter our source image via this new mask, and **generate background image without face.**



Finally, we **pad our new replaced faces** that store ahead one by one and get the new frame with all replaced face.



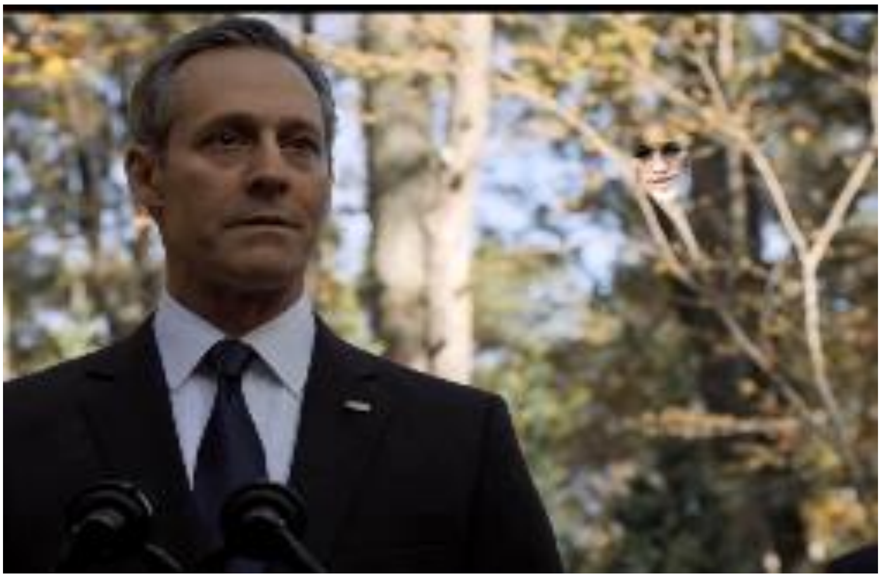


Apply the algorithm and procedure above to each frame extract from video, after dealing with all frames, we just combine them together and generate new video.

Challenge / Problems

The main problem we met in this project is **Dummy face problem.**

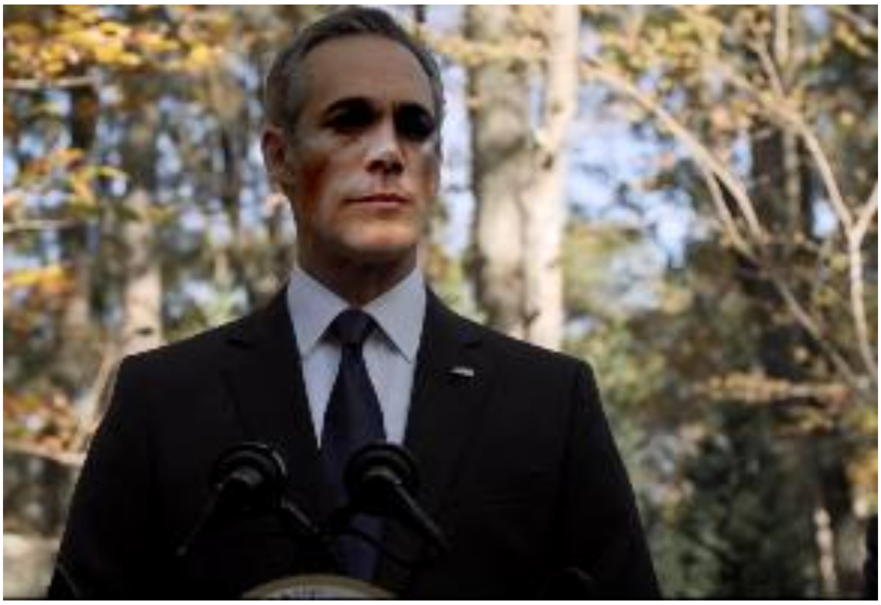
In face detection step, we use the 3rd library, Face ++, to achieve this function. However, for some images it will detect non – face region as face, and then gives us more feature points to confuse the algorithm. You can see a 'ghost face' on the right – top corner and unluckily we miss the true face.



For this problem, we came up with two possible solution.

First, for each iteration, we store our detected face mask region, only four corner points can achieve this, and make it to compare with previous mask. If the SSD of four corners larger than some certain threshold, we just drop it.

Second, since most dummy face region has less feature points than true face, so we choose to drop some detected faces with smaller number of feature points. But this solution has a drawback that it is likely to make us miss some side face.
Below is our new frame without dummy face.



Reference

1. Face ++ MATLAB API
2. Face Detection, Pose Estimation, and Landmark Localization in the Wild.
Xiangxin Zhu Deva Ramanan Dept. of Computer Science, University of California, Irvine