Project Report for CIS 581 (Option 2)
Computer Vision

Sagar Sinha SAGARSI@SEAS.UPENN.EDU
Harsh Verma HVERMA@SEAS.UPENN.EDU

Objective

The goal of this project is to detect and seamlessly replace face/faces in a given video using techniques like face detection, facial features detection, warping(TPS or affine) and Gradient Blending using Poisson Editing. We have chosen **Option 2**.

1. Different Experiments

We also tried Option 1 with various approaches to detect facial features by using techniques described in the sections below

1.1. Recognition and Segmentation using Shape Contexts

We created Code Book entries for various facial features namely left eye, right eye, nose and mouth using Shape Context. The points for which Shape Context histograms were calculated were selected by eliminating the closest points in the edge map of the various facial features. The code book entry that we stored consisted of $\{u_i\ \delta_i\}$. Here u_i is the Shape Context vector for feature i and δ_i is the distance of the center point of the feature from the center of the face. A sliding window scanned the whole test image calculating the Shape Context of the image patches selected by the sliding window, while storing the χ^2 distance of the code book entry with the Shape Context vector of the test image patch. Finally a cut off χ^2 distance was used to find the possible image patches which may contain a possible facial feature.The number of radial and theta bins, the outer and inner values of radius were chosen based on the thesis by Serge Belongie (2002) and their sample code to calculate Shape Contexts.

Problems: The main problem that we faced in this approach was the identification of too many false positives(approximately 30%) as facial features. We planned to use δ_i to eliminate false positives using the top down segmentation as described in (Wang, 2007), however we didn't use masks and weights in the code book entries which could be causing the large number of false positives.

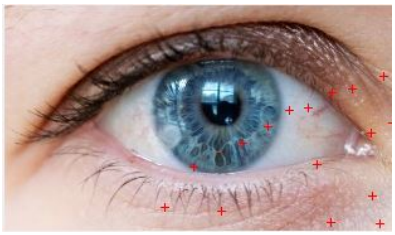

Figure 1. Points from where shape contexts were extracted

1.2. Pictorial Structure using HOG feature and SVM

We also experimented with Pictorial Structures where we first tried to identify different parts of the face using a classification algorithm(SVM) and then tried to minimize the cost(energy) of parts models matching as well as structural location matching. The parts were classified using SVM and the feature vectors were derived from MATLAB extractHOGFeatures() (Dalal & Triggs, 2005) by fixing the number of histogram bins ,which was achieved by using the same train image sizes for each of the features classes(eyes, nose and mouth).

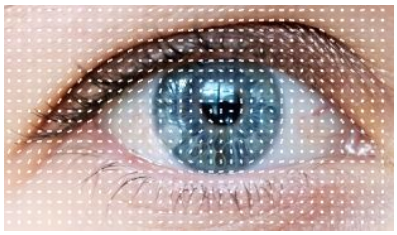

Figure 2. HOG Feature Visualization for Left Eye

Problems: Low number of training images led to the underfitting of learning model.We could not find a dataset consisting of just the facial features so we had to cut the facial features from the face images manually. Therefore we were not able to collect more training images and thus the classifier's performance was very poor which created a

problem for the first phase of pictorial structure.

1.3. Voting on result of Cascade Object Detector:

We also tried voting along with Matlabs Cascade Object detector in the Vision toolbox to detect faces and feature in images.We chose the face based on the most number of features detected and size of the bounding box.
For voting, Star-shaped model was used and feature detection was done by calculating the voted score for the center of face(nose).All features voted for the position of the center. We chose a nose based on consensus. Other features are chosen based on its distance from the center and alignment with other features.

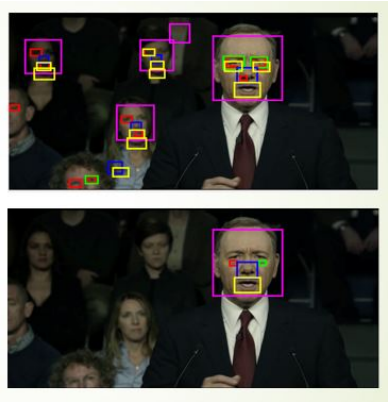

Figure 3. Voting removes the false positives within a face and also selects the best candidate for replacement. Magenta shows the face and red, green, blue and yellow show left eye, right eye, nose and mouth respectively

Problems - The results were satisfactory but it gave a very less number of feature points which were needed for a better TPS warping. The results from Voting and Face ++ are compared in the Comparison section.

2. Final Approach

After experimenting with different approaches as described above we finalized Face ++ for facial feature points/face detection, KLT for feature points tracking, TPS for warping the replacement face into the target face and Poisson Editing for Gradient Domain Blending. We also implemented Pose Detection and Motion Compensation for smoothing out the generated video. The approach comes under **Option 2**. The algorithm is described below.

2.1. Algorithm

Our pipeline uses the following methods:

1. $replaceFaceWIthFacePP$ - This is the main function.
2. $replacementFaceDict$ - construct a dictionary of replacement faces and corresponding feature points.
3. $getInterestPointsFPP$ detects faces in images and returns the facial features of one chosen face.
4. $pose\_detection$ returns the pose of the face, namely, facing left, center or right.
5. $wrapper\_replace$ - replaces the face in the target image with the replacement image
6. $stabilize\_video$ compensates the distortion between the frames in a given video.
7. For gradient blending we use a 3rd party library as it is faster than our implementation for Project 4A.
8. For TPS morphing we use our own code form Project 2.

2.2. Implementation

1. **Detecting facial feature points:** We use FacePlusPlus API to detect faces in images. Making simple HTTP requests to the API we get the face locations and other face attributes in response. We decide on which face to replace based on its proximity to the center of the image and the size of the detected face. We use the *83 facial feature points* for our purpose. This is shown in *Figure 4.*
2. **Choosing the replacement image:** *FacePlusPlus API* gives use the orientation of the image. We use the yaw angle to determine if the face if facing left, right or center. Accordingly, we choose our replacement face. This is shown in *Figure 9.*
3. **Tracking the feature points:** We use FacePlusPlus API to detect face and get the facial feature points once every 15 frames. For the intermediate 15 frames, *we use Matlabs Kanade-Lucas-Tomasi (KLT) algorithm* implementation to track the facial feature points detected in the 1st frame. This is shown in *Figure 14 and 15.*
4. **Morphing target image into the replacement image:** We use Thin Plate Spline morphing with warping fraction 1 and dissolve fraction 0 to morph the replacement image into the target image. The objective is to have the replacement image in the same size and shape of the target image but with the color of the replacement image. *Figure 5* shows this.
5. **Blending the resultant face in the target image** We use *Poisson blending* (Patrick Perez, 2003) to blend

the aligned morphed image onto the replacement image using a binary mask constructed from the convex hull of the facial feature points. The mask is shown in *Figure 6*.We smooth the blending boundary around the mask region using a Gaussian Filter. The blended and the smoothed image are shown in *Figure 7 and 8 respectively.*

6. **Motion Compensation:** To correct the distortion between two frames we morph the neighboring video frames to get a stable video. We achieve this by determining the affine image transformations between the frames using Matlabs estimateGeometricTransform and apply it to the point correspondences between the images. We use Fast Retina Keypoint(FREAK) descriptor around the corner points and MSAC as our consensus algorithm to match the point correspondences. This is described and visualized in detail in *Section 3.2.*

2.3. Visualization/Analysis

1. Detection of facial features and the face

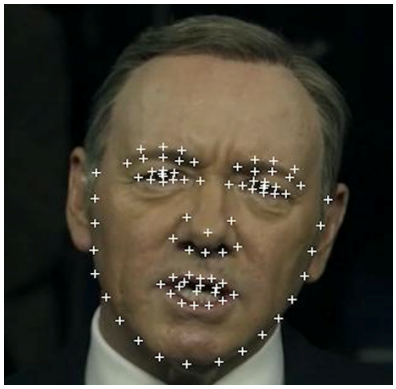

Figure 4. Detecting Facial Feature Points

2. Morphing the replacement image to target image

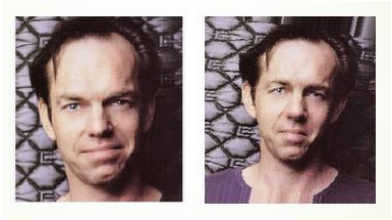

Figure 5. Replacement Face Morphed into the Target Face

3. Selecting the Mask

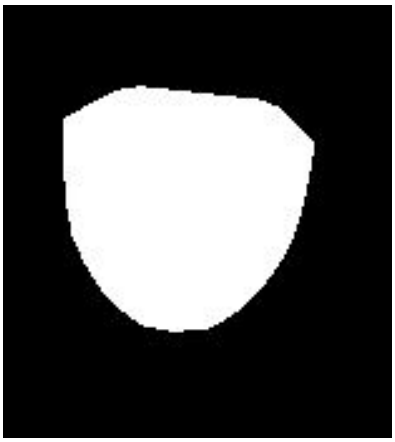

Figure 6. Mask for cutting and pasting the replacement face

4. Poisson Editing

Figure 7. Result of Poisson Blending of the replacement face into the target face using the mask shown above

5. Smoothing the blended image

Figure 8. Final Smoothed Blended Image

3. Extra Credits

3.1. Pose Detection

We used replacement pictures in different poses

Figure 9. Different Poses Replacement Pictures

Based on the yaw angle of the target face the appropriate replacement picture was chosen.The yaw angle gives the direction of the face in the z-axis. Yaw angle is one of the Euler's angle describing the orientation of the object in space. Face++ gives the Euler angles through its API.

Figure 10. Replaced Target Faces with different poses based on Yaw Angle.

3.2. Motion Compensation

Motion Compensation is smoothing out the movement of pixels between the consecutive frames in a video. Due to the TPS warping of the replacement face into the target frame, sometimes there is an abrupt movement in some part of the video frame. To resolve this we used Motion Compensation by warping one frame into the other frame for a seamless and uniform transformation. The process is as follows:

1. Fetch every two consecutive video frames.
2. Find the corner points in the two frames using FAST(Features from accelerated segment test) corner detection algorithm.It is faster than other corner detection algorithms
3. Find the Fast Retina Keypoint (FREAK) descriptors for each of the corner points.
4. Match the descriptors to the corresponding descriptors in the other frame. This gives a correspondence between the feature points of the two frames.
5. Find inliers using MSAC (same as RANSAC but it also takes into account the quality of the consensus by using Likelihood probablity).Then find a geometric transform using these point correspondences and apply these to all the pixels in the first frame to get the pixels in the second frame. Intuitively this finds the transform from the background pixels and applies them to the flickering points in the frames. Figure 11, 12 and 13 show this process.

Figure 11. Previous Frame

Figure 12. Next Frame

Figure 13. A and B show the matched points in the two frames. As can be seen here background frames coincide.A slight shift in the matching corresponding points can be seen around the eyes and nose

3.3. Tracking

Tracking was also done to predict the position of the control points in the subsequent frames. This reduced the number of times the feature points were detected in the video and also smoothed out the movement of the control points through the frames giving a more seamless experience. Tracking was done using the Kanade-Lucas-Tomasi (KLT) algorithm on subsequent 15 frames after the feature

points were detected. Figures below demonstrate Tracking:

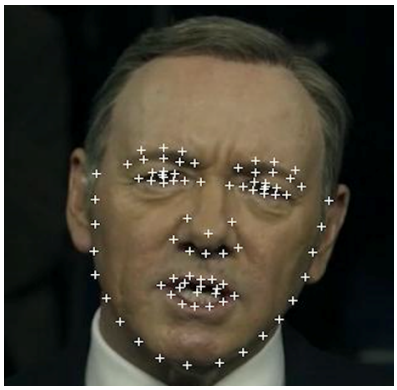

Figure 14. Original Control Points

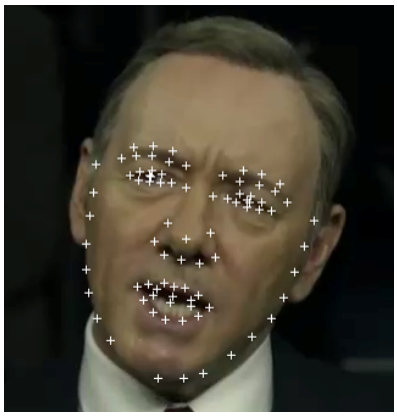

Figure 15. Control Points generated by KLT based on the last frame as can be seen KLT loses some of the Control Points

4. Comparison

Feature points retrieved from MATLAB Cascade Object Detector after applying voting(Section 1.3) are shown below in *Figure 16* which can be compared with the Feature Points obtained from Face++ as shown in Figure 14:

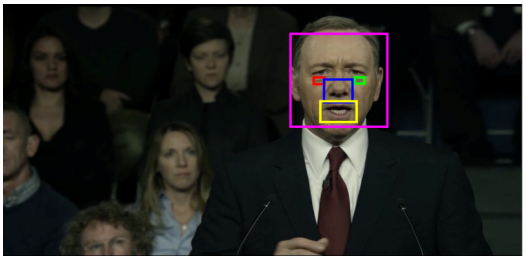

Figure 16. As can be seen this approach only gives 4 control points whereas Face++ gives 83 control points as shown in Figure 14

References

Dalal, Navneet and Triggs, Bill. Histograms of oriented gradients for human detection, 2005.

Patrick Perez, Michel Gangnet, Andrew Blake. Poisson image editing. Technical report, Microsoft Research, UK, 2003.

Serge Belongie, Jitendra Malik. Shape matching and object recognition using shape contexts. Technical report, 2002.

Wang, Shi, Song I-fan Shen. *Object Detection combining Recognition and Segmentation*. PhD thesis, Department of Computer Science, University of Pennsylvania, 2007.