Project 4B - Face Replacement (Option 1)

Group No. 15 :

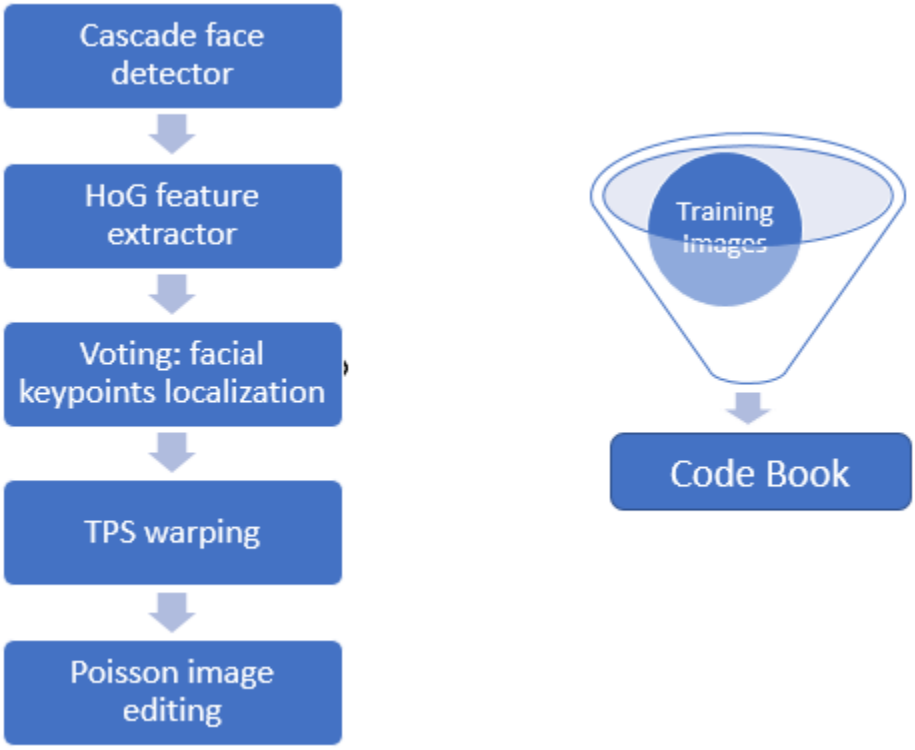
1) **Harshal Godhia**
2) **Bhairavi Mehta**

Objective :

To identify faces (target faces) in videos and replace them with a face from a predefined database of faces (we call one such face the replacement face). This has many industry applications such as fading or swapping one or more faces when capturing video before releasing to public to protect privacy of people captured in video and images such as Google Street view.

In our implementation, we have focused on identifying facial features such as eyes, nose and mouth in the target face using the Voting method [1].

Algorithm :



The basic pipeline for face replacement for a single frame of the video is as shown in the above figure.

Steps for face replacement :

1) Generate a codebook that describes the various facial features such as eyes, nose, mouth and facial boundary using HOG features extracted from the training images.
2) Select the corresponding control points in the replacement image.
3) Select the mask for the replacement image
4) Detect the face to replace in the first frame of the target video using Viola-Jones algorithm
5) Identify facial features to track
6) For each frame of the target video,
   a) Find the new position of the detected face using KLT algorithm
   b) Extract HOG features
   c) Find top k matches of each extracted feature from the codebook using chi-square distance
   d) Detect the facial keypoints such as eyes, nose, mouth and facial boundary using Voting
   e) Warp the replacement face to the target face using Thinplate spline warping
   f) Replace the warped face in the target frame and perform blending using Poisson Image editing

Implementation :

Code book generation

The code book summarizes the information contained in the training images. Our code book comprises of the following data structures stored in the codebook.mat file

x_coord, y_coord : X, Y coordinates of the center of the facial features	codebook : HoG feature descriptors of the facial features centered at the corresponding X,Y positions	bounding_box : Properties of the bounding box that encloses the face i.e(upper_left_x,upper_left_y,width,height)

Since we have 14 images, the entire data structure would comprise of (14*12=168 rows of the above format). The 12 features correspond to 3 points on the hairline from left to right, 5 points on the face boundary from left to right, the left eye, the right eye, the nose and the mouth.

Note: above the X, Y positions are manually selected using MATLAB freehand and point selector from the training images.

Facial keypoint detection in replacement image

The facial keypoints and the replacement mask for the replacement image have been manually selected using Matlab's ginput() and imfreehand() functions.

Face Detection

We have detected the faces in each frame using Matlab's built-in Cascade Object Detector [3]. The face to be replaced has been manually selected from the multiple faces returned by the detector.

Face Tracking

Face tracking in successive frames has been performed using KLT algorithm [2].

Facial key point localization

- *Generating top-k matches*
  At a high level for the face detected in the video frame we run a sliding window in the bounding box of that detected face and perform a chi-squared comparison of the HoG feature vector of the sliding window against each of the HoG vectors in our code book. We then generate the top 10 matches. Note we proportionately alter the sliding window size to account for face size differences.

  $$\text{Sliding window size} = (cell\ size) * \frac{\text{current detected bounding box}}{\text{corresponding training image bounding box}}$$

- *Voting*
  We now create and maintain a voting map for each of the 12 key features - 3 points on the hairline, 5 points on the face boundary, left eye, right eye, nose and mouth which will help us pinpoint the facial feature location coordinates. For every match to the code book, we calculate the approximate position of the remaining 11 features and store the e^-(chi-square distance metric of the current feature) at these 11 locations in the corresponding feature maps. To account for precision and variability we blur the 5*5 patch at each of these locations in the corresponding feature maps using a 5*5 gaussian filter.

- *Locating the feature coordinates*
  To ensure robust results in each map we find the mean score and use that as a threshold to ignore values lower than that. Finally we output the pixel location in each map that is the centroid of the thresholded map as the location of that feature for the detected face.

Facial warping and blending

We have used the previous code implementations for TPS warping and Poisson Image Editing.

Visualization/Analysis

Fig1: Training image set

Fig2: HoG (histogram of gradients) identified for facial features in one of training images

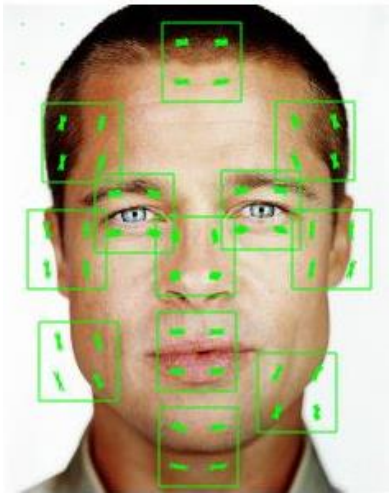


Fig 3: 12 facial key points localized in target face on test/video frame

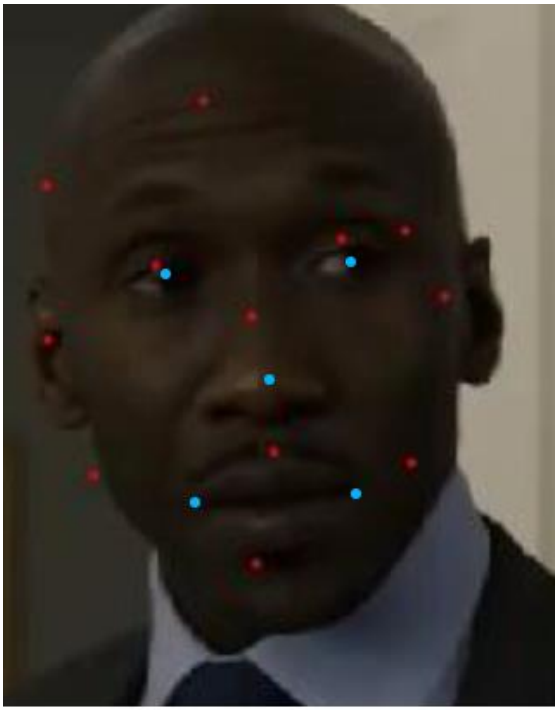
Fig 4: Replacement face

Fig 5: Corresponding frame with face replaced

Comparison

We can compare our results with a state-of-the art commercial face detection and key point localization software (http://www.faceplusplus.com/demo-landmark/). In the below figure the blue dots correspond to those generated by the 3rd party software (Face++) while the red dots represent our implementation. As one may notice our results are very similar.

Fig 6: Comparison of our implementation with 3rd party software (Face++)



References

[1] L. Wang et al. "Object detection combining recognition and segmentation." ACCV 2007, https://pdfs.semanticscholar.org/9c0d/7e4c9e4b0d9a2ffa9b4a359034773949c98c.pdf

[2] Face tracking using KLT algorithm, https://www.mathworks.com/help/vision/examples/face-detection-and-tracking-using-the-klt-algorithm.html

[3] Face detection using Viola-Jones algorithm, https://www.mathworks.com/help/vision/ref/vision.cascadeobjectdetector-class.html