Mask-it

Dhaval Popat (dkp288), Sonit Samal (ss11354), Vikram Sunil Bajaj (vsb259), Yash Panchamia (ygp210)

1. MOTIVATION

We started the course in Computer Vision with very captivating topics such as Edge Detection and Orientation, Gaussian Smoothing, Affine transformations, and many more. Equipped with the skills in using these and applying the knowledge towards practical use, we came across the various filters being used across social networks to make visual interaction and communication more enjoyable. Inspired by the dynamic masking of the faces with images and animated figures achieved by famous applications like Instagram and Snapchat, we decided to take up the task of designing a system capable of detecting a human face in an image, determining landmarks on it, mapping and morphing of a suitable mask and overlaying the mask.

2. APPROACH

We have divided our project into four modules as follows -

A) Face detection
B) Facial landmark detection
C) Alignment of the mask
D) Overlaying the mask to facial image

A) FACE DETECTION (Implemented by Yash Panchamia - ygp210)

The approach followed in this module relied based on Histogram of Oriented Gradients (HOG). Since, our main motivation and objective from the beginning of the project was to code everything ourselves, also using the concepts taught to us during the course and without using the existing libraries and/or machine learning algorithms, we relied on HOG because it gives a strong set of features for the human face. HOG is a feature descriptor used to detect objects, in our case, it was used to detect the human faces. The HOG descriptor technique counts the occurrences of gradient orientation in localized portions of an image - detection window, or region of interest (ROI). Implementation of the HOG descriptor algorithm is as follows:

1. Divide the image into small connected regions called cells, and for each cell compute a histogram of gradient directions or edge orientations for the pixels within the cell.
2. Discretize each cell into angular bins according to the gradient orientation.
3. Each cell's pixel contributes weighted gradient to its corresponding angular bin.
4. Groups of adjacent cells are considered as spatial regions called blocks. The grouping of cells into a block is the basis for grouping and normalization of histograms.
5. Normalized group of histograms represents the block histogram. The set of these block histograms represents the descriptor.

Computation of the HOG descriptor requires the following basic configuration parameters:

1. Masks to compute derivatives and gradients
2. Geometry of splitting an image into cells and grouping cells into a block
3. Block overlapping
4. Normalization parameters

So, for the first part of the HOG, we divided the image into small connected cells and then for finding the x and y gradients, we used the Sobel operator which we previously learnt to use during the course. We also calculated the gradient magnitude and gradient angle. We then followed the algorithm to discretize each cell into bins according to the gradient orientation was found by using the Sobel operator. As a result, a HOG descriptor was obtained which

was not up to the mark. Hence, we decided to move forward with a more accurate approach of Viola Jones algorithm for face detection.

Viola Jones algorithm describes a face detection framework that is capable of processing images extremely rapidly while achieving high detection rates. There are three key contributions. The first is the introduction of a new image representation called the "Integral Image" which allows the features used by our detector to be computed very quickly. The second is a simple and efficient classifier which is built using the AdaBoost learning algorithm to select a small number of critical visual features from a very large set of potential features. The third contribution is a method for combining classifiers in a "cascade" which allows background regions of the image to be quickly discarded while spending more computation on promising face-like regions. The algorithm has four stages:

1. Haar Feature Selection
2. Creating an Integral Image
3. Adaboost Training
4. Cascading Classifiers

A few properties common to human faces:

- The eye region is darker than the upper-cheeks.
- The nose bridge region is brighter than the eyes.

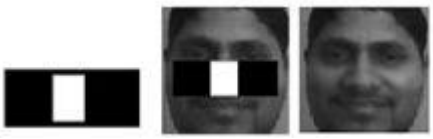
Composition of properties forming match able facial features:

- Location and size: eyes, mouth, bridge of nose
- Value: oriented gradients of pixel intensities

The four features matched by this algorithm are then sought in the image of a face (shown at right).

Rectangle features:

- Value = Σ (pixels in black area) - Σ (pixels in white area)
- The difference in brightness between the white and black rectangles over a specific area
- Each feature is related to a special location in the sub-window



Haar feature that looks like the bridge of the nose is applied onto the face

Haar feature that looks like the eye region, which is darker than the upper cheeks, is applied onto the face

B) FACIAL LANDMARK DETECTION (Implemented by Dhaval Popat - dkp288)

The approach followed in this module relies based on segmenting skin region from the prominent facial features like eyes, lips, and nose. The skin region is detected using skin detection in YCbCr and RGB color spaces.

The image is converted to YCbCr color space using the following equations[1]

(i) Y = 0.257*R + 0.504*G + 0.098*B + 16
(ii) Cb = 0.148*R - 0.291*G + 0.439*B + 128
(iii) Cr = 0.439*R - 0.368*G - 0.071*B + 128

A pixel is a skin pixel if it satisfies both of the following equations[2]

(i) Y > 80 AND 85 < Cb < 135 AND 135 < Cr < 180
(ii) (R>95) AND (G>40) AND (B>20) AND
(|R-G|>15) AND (R>G) AND (R>B) AND (Max{R,G,B} - Min{R,G,B} > 15)

The skin pixels are assigned grey level 0 and remaining pixels are assigned grey level 255. Once all the skin pixels are segmented from the facial image, all the facial features that don't contain skin such as eyes, nostrils, and lips are distinctly visible. The noise is removed, and these features are enhanced using Gaussian smoothing of kernel size (15,5), and then a combination of binary and OTSU thresholding. The horizontal kernel is used because most of our facial features are typically horizontal in nature. Since the input image to this module consists of only a face, we divided this image into three different parts; upper left face consisting of left eye, upper right face consisting of right eye, and lower face consisting of nostrils and lips. Refer to these images in the results section for better understanding.

The idea of extracting eyes and lips from this processed image relies on the fact that eyes and lips are horizontally wide. This means that when we project the frequency of white (i.e. non-skin) pixels for each row of the image, the row containing eyes and lips will ideally have highest frequency. But it is possible to have cases where some other rows would be having greater frequencies than the rows consisting of eyes and lips. The rows with all white (non-skin) pixels are removed from this projection, as these rows are not a part of the input face. This helps in removing the false highest frequency rows. There still might be some false highest white frequency rows left which will be handled as explained below.

The image is traversed in sorted descending order of white (non-skin) pixels. This means that the row with most white pixels is traversed first with the goal of detecting eyes. The program starts looking for white pixel (i.e. start of the left eye), since we know that all white pixels will be in a sequence for each eye and lips. We have kept a margin error of one percent of total pixels in a row. This means that if width of the row containing eyes is 1000 pixels, it is okay if there are less than 10 black (skin) pixels detected in between white pixels. After a point is reached where there are more than 10 pixels are black, we mark this pixel as end of the left eye. To know if we have really found the eyes or some noise, a validation is performed where we check if the size of this detected eye (end_pixel-start_pixel) is greater than 20 percent of total width of image. If not, we discard this eye and traverse through next row with second highest white pixels, and so on. This removes almost all the false positives and accurately detects the left eye.

Similarly, the upper right image of the face is traversed to detect the right eye. The lower face image is traversed to detect the lips. A similar approach is used for lips as well, but the minimum size is increased to 25 percent of the width of the image. Once the lips are detected, the lower half of the face image is cropped in such a way that only the nostrils feature is highlighted with white pixels. This is done using the following mapping: x1, y1 = 0, start-lips[y-coordinate] and x2, y2 = end-lips [x-coordinate], end-lips [y-coordinate]. After the image is cropped, the left and right nostrils are being detected by traversing from the top left of the image. We are then finding the first white pixel and marking it as the left nostril. Once the left nostril is detected, the traversing now starts from right of the image at that row itself. If another white pixel is found, it is validated and marked as right nostril. Otherwise, the left nostril is reset to not found, and the program continues from next row. The validation applied here is absolute distance between the two nostrils should be greater than 20 percent of the width of cropped image consisting the nose feature.

C) ALIGNMENT OF THE MASK (Implemented by Vikram Sunil Bajaj - vsb259)

This section deals with morphing the mask to match the landmarks detected in the previous stage.

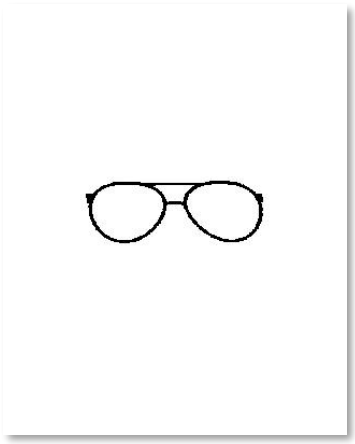
The following two masks have been used, and their landmarks were manually annotated as shown below:

[1] Diedrick Marius, Sumita Pennathur, and Klint Rose: Face Detection Using Color Thresholding, and Eigen image Template Matching.
[2] Alireza Rahmani Azar and Farhad Khalilzadeh: Real time eye detection using edge detection and Euclidean distance.

The following face image is used (green circles denote eye landmarks):

STEPS

1. The masks have a transparent background, so their background is first made white
2. The mask image is then padded with 255s (white) to match the size of the face image
3. After padding, the landmark locations are updated to match their positions before padding
4. Now, a transformation matrix is computed to be able to transform the mask landmarks to match the facial landmarks from the previous stage (for the glasses mask, the eye landmarks were used, whereas for the moustache mask, nose and lips landmarks were used)
   The image below shows the morphed glasses mask:



5. The next and final stage deals with overlaying the morphed mask onto the face image

D) OVERLAYING THE MASK TO FACIAL IMAGE (Implemented by Sonit Samal - ss11354)

Overlaying one image with another involves two steps:

i. Image Resizing
ii. Alpha Blending

Image Resizing

Image resizing requires the coordinates of the smaller image to be mapped to the coordinates of the larger image. This is achieved by Image interpolation. Image interpolation works in two directions and tries to achieve a best approximation of a pixel's intensity based on the values at surrounding pixels. This is achieved by using certain algorithms like nearest neighbour, bilinear transformation and Lanczos. For our purpose, we have used Lanczos interpolation.

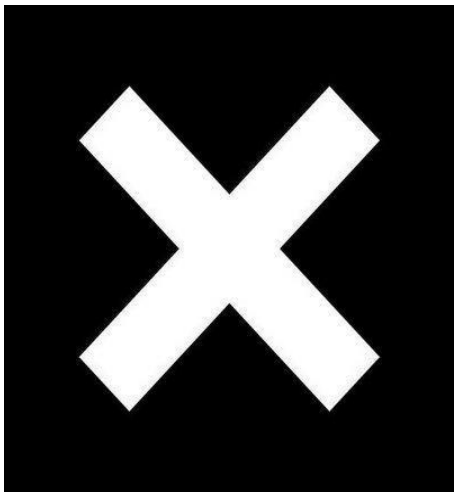
Lanczos resampling uses a convolution kernel to interpolate the pixels of the input image to calculate the pixel values of the output image. The Lanczos convolution kernel k(x) is defined as:

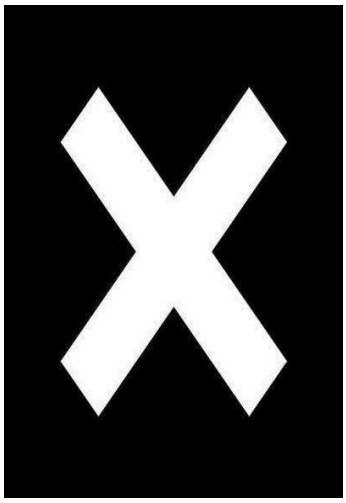
$$k(x) = \begin{cases} \text{sinc}(x).\text{sinc}(x/a) & ; \text{if } |x| < a \\ 0 & ; \text{otherwise} \end{cases}$$

where a=3 in the present implementation.

Lanczos interpolation has the tendency to preserve the alpha value of the pixels, something which plays a very important role in the next part of the process, Alpha blending. The interpolation method is much like cubic/bilinear interpolations, Lanczos interpolation also helps to keep the edges smooth, sometimes increasing the contrast. Benefits of Lanczos interpolation also includes the handling of detailed graphics.

Image 1 Image 2 Resized Image





Alpha Blending

For overlaying, we followed a process called Alpha blending, wherein, the foreground image is selectively made transparent and then is pasted over the background image with corresponding coordinates. The formula used for Alpha Blending is:

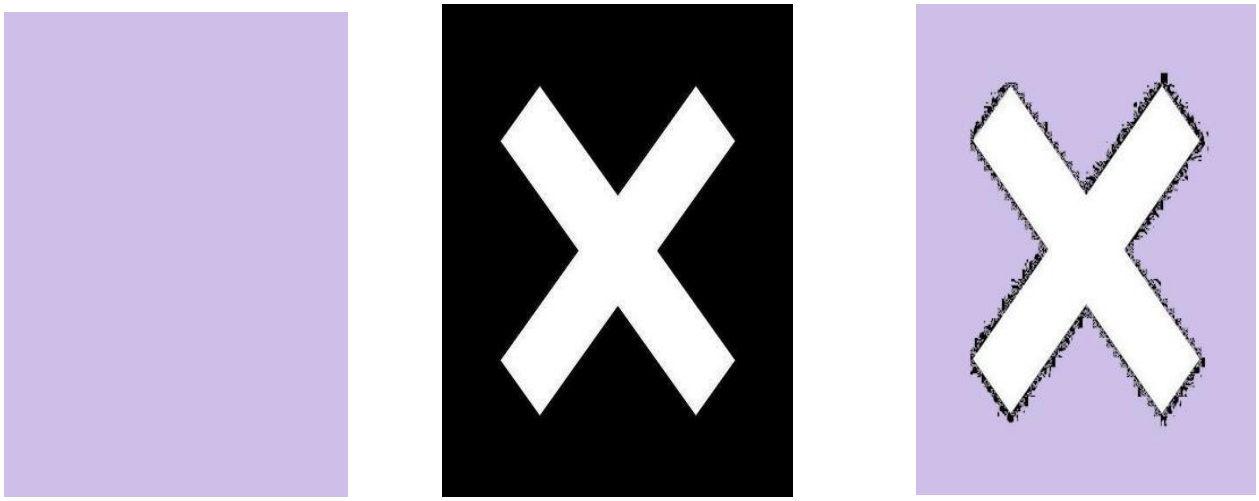
Image = alpha * (Mask) + (1-alpha) * (Background)

The value of alpha is determined by using an Alpha mask which is obtained by segregating the Alpha channel from the read PNG image.

The transparent region is then identified, and its alpha value is changed to 0. The modified matrix is then normalized by dividing with 255, to keep the values in the range of (0,1). After the normalization, the alpha mask is multiplied with the mask image matrix.

The complement of the alpha mask is determined, and the background is blended along with it. The results from both the parts are then added to make the final image, which contains the foreground mask mapped to the landmarks of the background and blended over it.

Background Image | Foreground Image | Alpha Blended Image



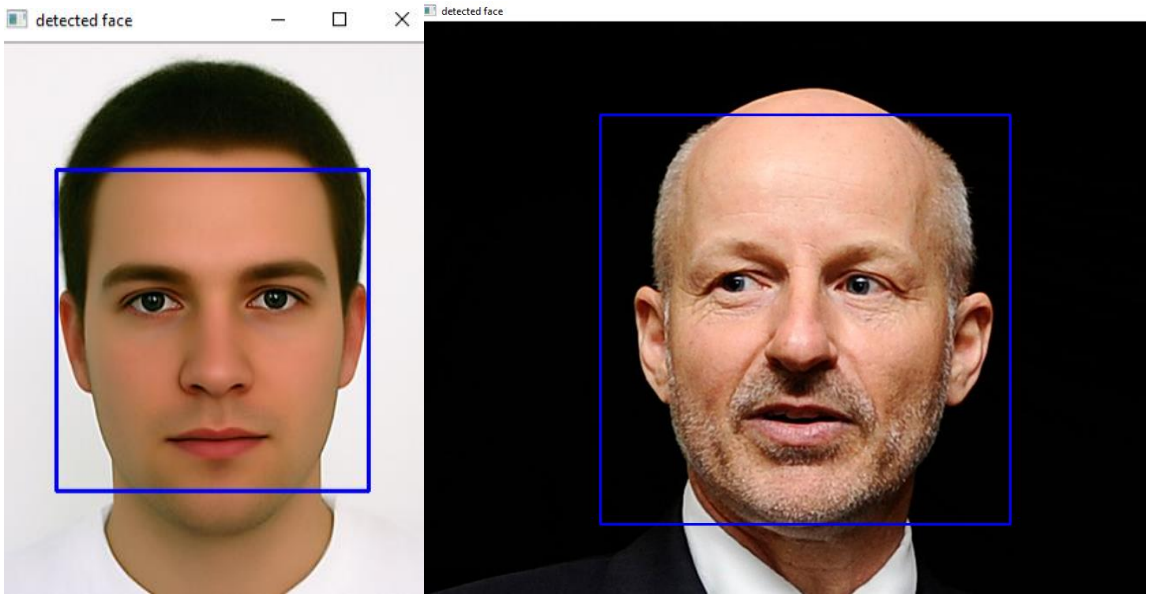
3. DATA USED

For this project, since we wanted this project to be an image processing project, we don't have as such a dataset (example training testing data). We have used a few people's faces images for testing. We also have used facial masks (like Snapchat filters). Since, the first approach that consisted of HOG wasn't so efficient for face detection, we have used Viola Jones's haarcascade_frontalface_default too.

4. RESULTS AND CRITICAL DISCUSSIONS

A) FACE DETECTION (Implemented by Yash Panchamia - ygp210)

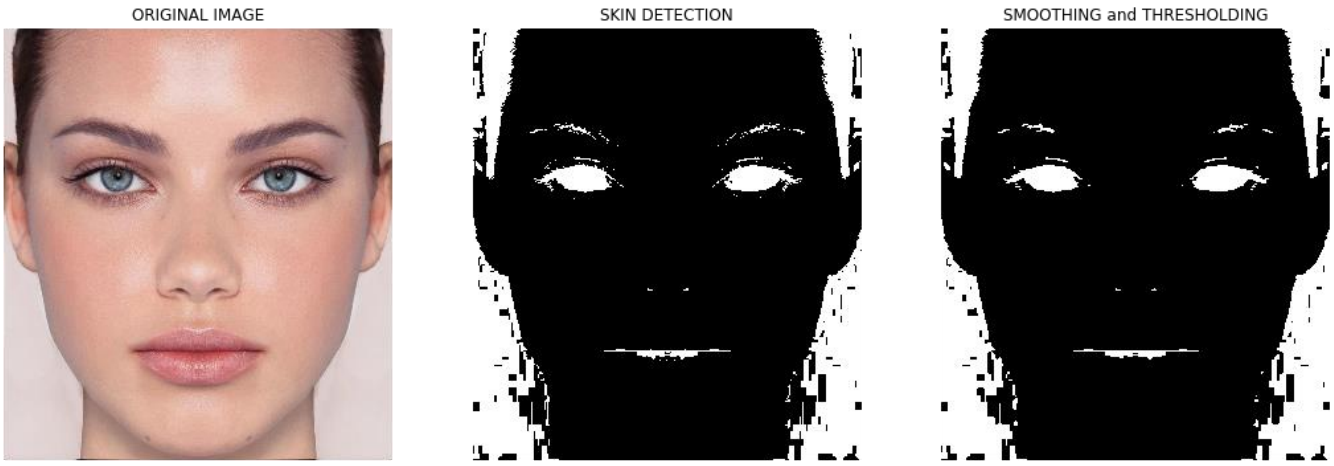
Here, we have used Viola-Jones algorithm and obtained the following output:



B) FACIAL LANDMARK DETECTION (Implemented by Dhaval Popat - dkp288)

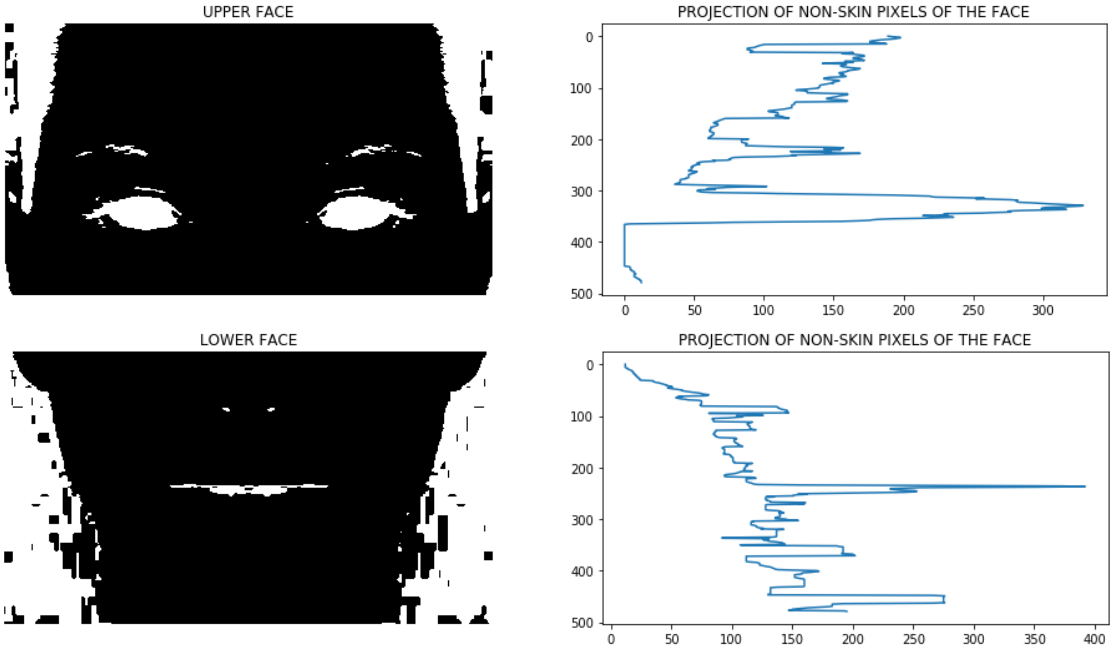
The idea of this module is based on segmenting skin region from facial features. Some questions may arise regarding robustness of this approach. Let's address them one by one and see if we get expected results.

What if the background is similar to the skin color?

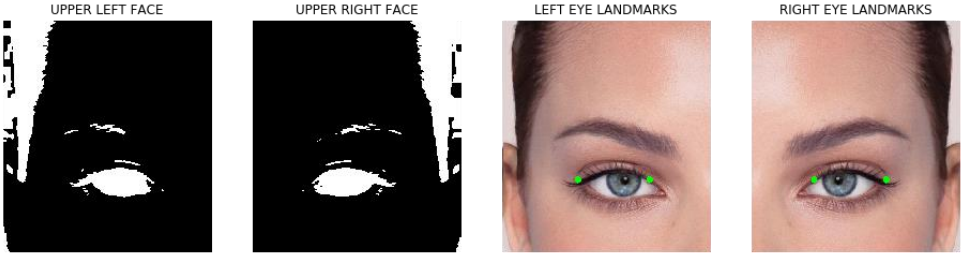


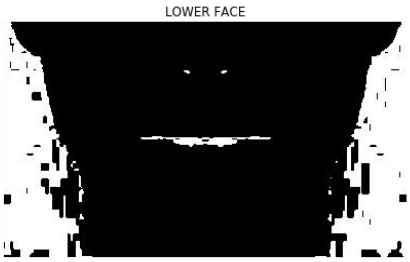
It can be seen from the above results that even though the background has similar color as that of the skin, the algorithm has accurately segmented the skin region and replaced them with black pixels. The prominent features like eyes, nostrils, and lips have white pixels.

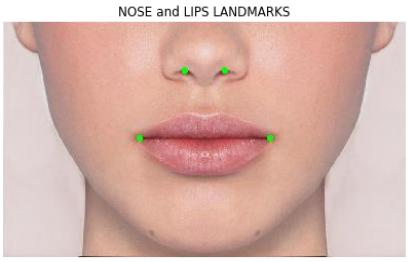
Let's now project the frequency of white pixels for each row.



It can be clearly seen that the projection of eyes and lips have maximum frequency. But it is possible to have some other rows with highest frequencies. The techniques that have been used to overcome this challenge have been discussed above in the 'APPROACH' section.

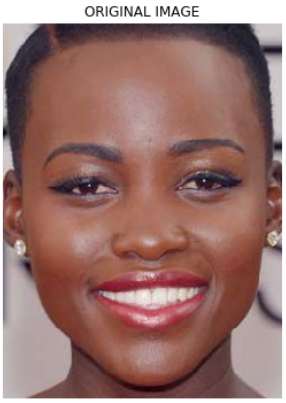




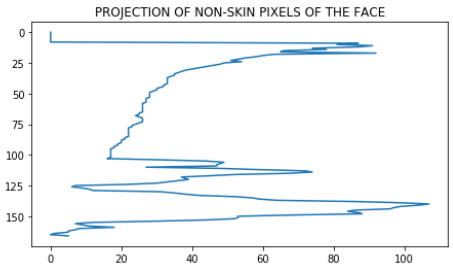


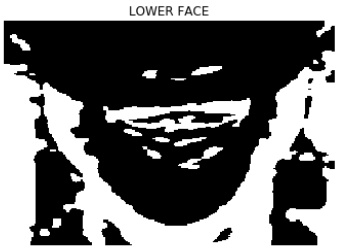
It can be observed that eyes, nostrils, and lips have been detected accurately. Although the points on the eyes are not the best fits, it works well within the scope of our application.

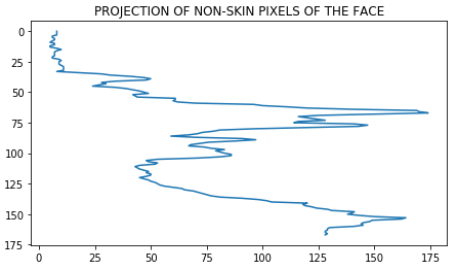
Does this work for varied skin tones? Let's try it on another image with a bit darker skin tone.



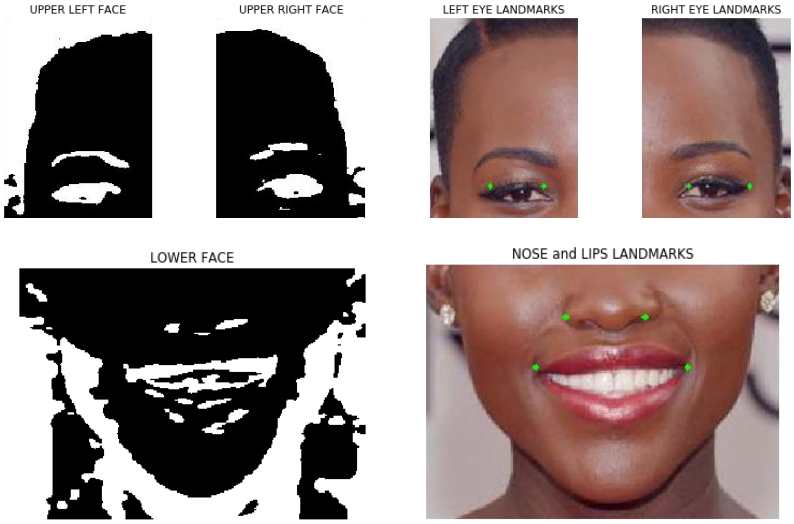
The algorithm does a good job in segmenting the skin region, but it could have been better by using more sophisticated techniques involving CNN's and similar approaches. But it looks better after smoothing and thresholding, the noise has been removed and the image looks neat.





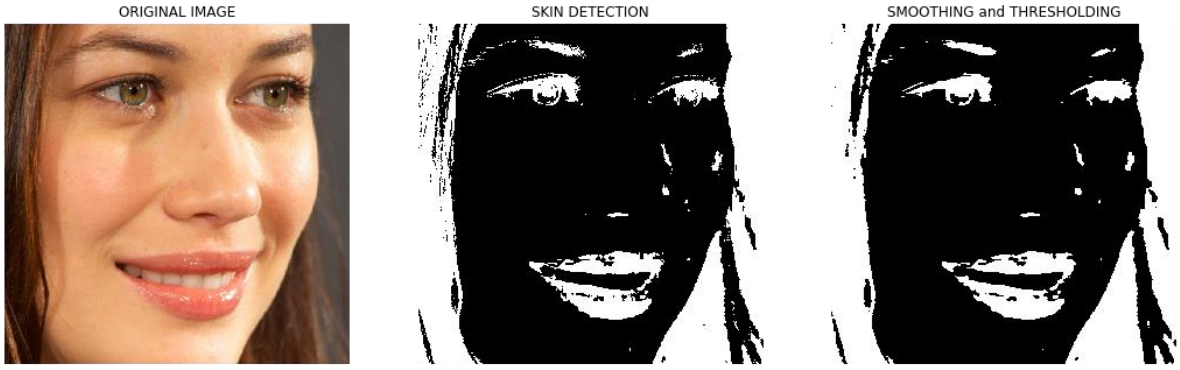


The projections of the upper face image have 0 frequency in the projection because it is completely white thus discarded as not being a part of the face.

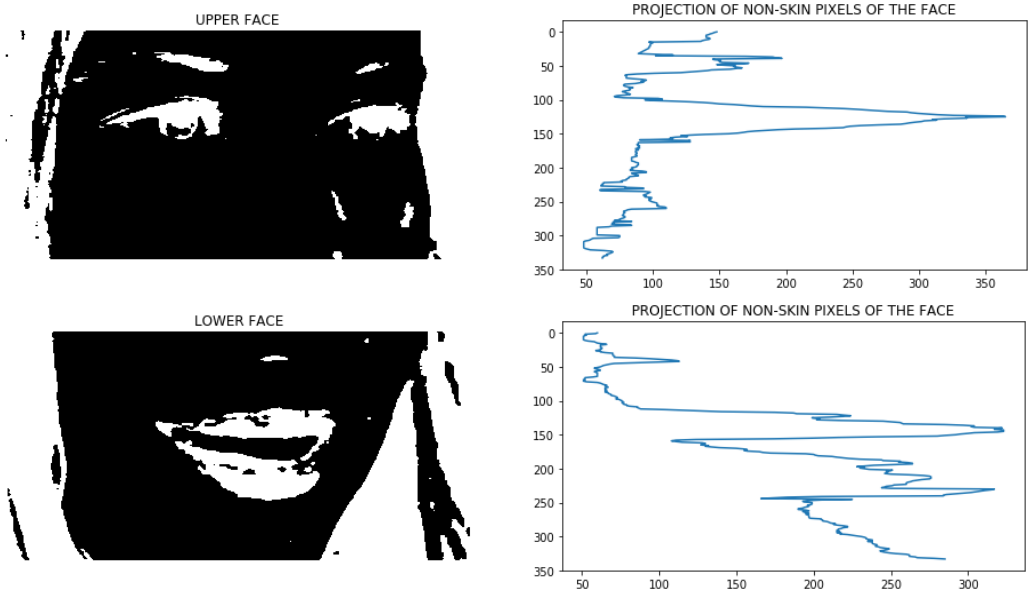


The landmark points detected are fairly accurately for this image as well.

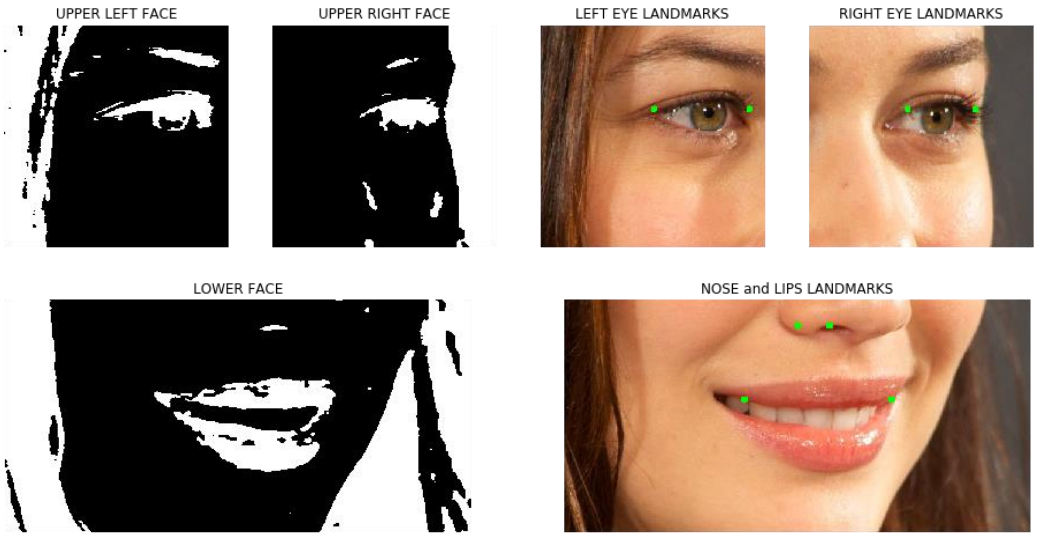
What if the image is not a frontal face and is sideways instead?



The skin segmentation is done very well. But the nose is not proper because of the side-way looking picture.



The projections are done properly. It can be seen that the eyes and lips have maximum frequency of white pixels.



It can be observed that the detection nostrils are not perfect, since one of them is not visible.

C) NON-LINEAR ALIGNMENT OF MASK (Implemented by Vikram Sunil Bajaj - vsb259)

The results for this section have already been discussed earlier.
This section mainly involved the warping of the mask based on the facial landmarks, so as to allow the mask to fit properly onto the person's face.

As discussed earlier, the glasses mask was warped to fit points on the face, based on the eye landmarks.

D) OVERLAYING THE MASK TO FACIAL IMAGE (Implemented by Sonit Samal - ss11354)

Background Image

Foreground Image

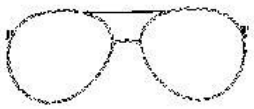
Resized Image

Outputs of Alpha Blending:

The results of applying alpha blending are shown below.

Background Image

Resized Image



Alpha Blended Image

5. FUTURE WORK

We can have pretty good future scope for this project.
For the future work, we can extend this project to:

1. Work with multiple faces in an image.
2. Perform real-time face detection and mask overlaying
3. Apply more masks (example beard, animal ears etc)