# DIGITAL FACE MAKEUP BY EXAMPLE

*Siddharth Khera (2015MT60567)*
*Ankur Sharma (2015CS50278)*

**AIM :** Given a reference makeup image, transfer the makeup from the example image to the subject image.

**ASSUMPTIONS:**

1. Subject/Example Photo should have exactly and only 1 face. This won't work on multiple faces (if detected).
2. No extended moustache/beard should be there. This will only work on some selected facial components.
3. Gradient Editing done by Image Pyramid method.

**PROCEDURE AND PIPELINE**: Given below is the pipeline followed for transferring the makeup from the reference image to the subject image.

## STAGE 1: FACE ALIGNMENT

We used a *dlib-64 bit face detector trained file* to get the control points on the face. To mark the hair boundary, we adopted a manual strategy in which we ask the user to mark the 10 boundary points on the upper forehead. Facial components defined by control points includes eyebrows, eyes, nose, nostrils, lips, mouth cavity, and other facial skin, which are further divided into three classes, 'face skin' as C1, 'lips' as C2, and 'eyes and mouth cavity' as C3.

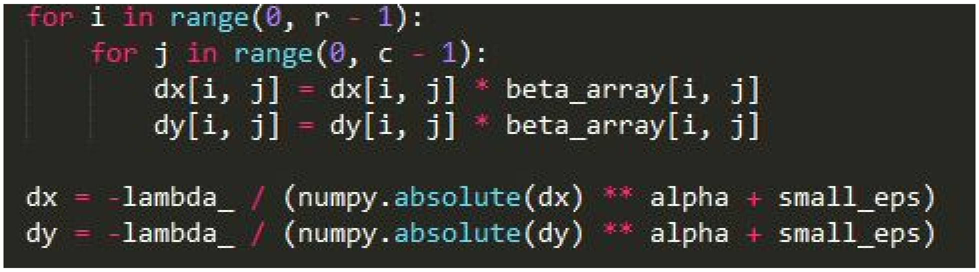
Warping is done using Delaunay triangulation. This is the default implementation taken from python-scipy library.

Forehead control points are selected manually on the face from left to right in the clockwise direction. Note that the order in which the points are selected are crucial, so please make sure that you select points one by one in the clockwise direction on the forehead-hair separation.

## STAGE 2: LAYER DECOMPOSITION

Using CIELAB color space, we first separated this into lightness(L*) and color channel(a*, b*). To further separate the lightness layer into face structure and skin detail layer, we first obtained the structure layer by applying a modified form of WLS filter on L* channel of the image. Original WLS filter does a uniform smoothing across the entire image, but we needed differential smoothing for different parts on the face. So, while minimizing, we multiplied each pixel by some *beta(p)* which varies smoothly from region to region. This smooth variation is further achieved using a Gaussian function whose value decreases as the distance from the pixel increases.

WLS standard implementation has been taken from *shiruilu* . To modify the same, we multiplied corresponding values with the beta value at that point as shown below.



```
for i in range(0, r - 1):
    for j in range(0, c - 1):
        dx[i, j] = dx[i, j] * beta_array[i, j]
        dy[i, j] = dy[i, j] * beta_array[i, j]

dx = -lambda_ / (numpy.absolute(dx) ** alpha + small_eps)
dy = -lambda_ / (numpy.absolute(dy) ** alpha + small_eps)
```

### STAGE 3: SKIN DETAIL TRANSFER

The resultant skin layer is a weighted sum of the two skin layers of the reference and example image. So, as mentioned in the paper we set Delta-i = 0, and Delta-e = 1, so as to copy the entire skin detail layer of the example image to the resultant image.

### STAGE 4: COLOR TRANSFER

This is achieved using alpha-blending of the two color layers of ‘a’ and ‘b’ with the value of gamma = 0.8.

### STAGE 5: HIGHLIGHT AND SHADING TRANSFER

Here, we adopted a different strategy. We used the Gaussia_down and Gaussian_up methods on the original image, and added this to the corresponding laplacian of the original image triangle by triangle.

### STAGE 6: LIP MAKEUP

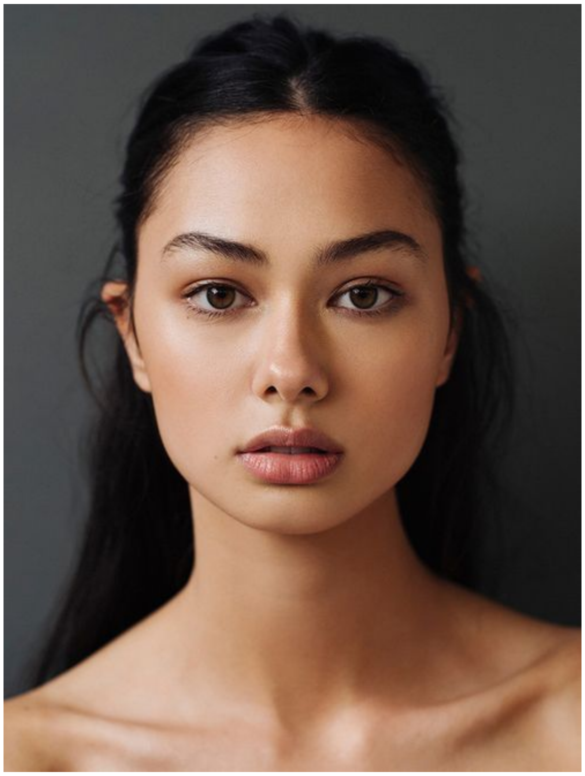
In lip makeup part, I need to fill each pixel of resultant image from reference image guided by the subject image. Basic idea is for each pixel, I need to find the corresponding pixel which is as close and as similar to that pixel. Minimization of the term yields the adequate position of the pixel to fill the L* channel of the resultant image with. Similarly, we can fill the a* and b* channel using this pixel which we computed earlier using the minimization of this term.

### PRECAUTIONS & LIMITATIONS:

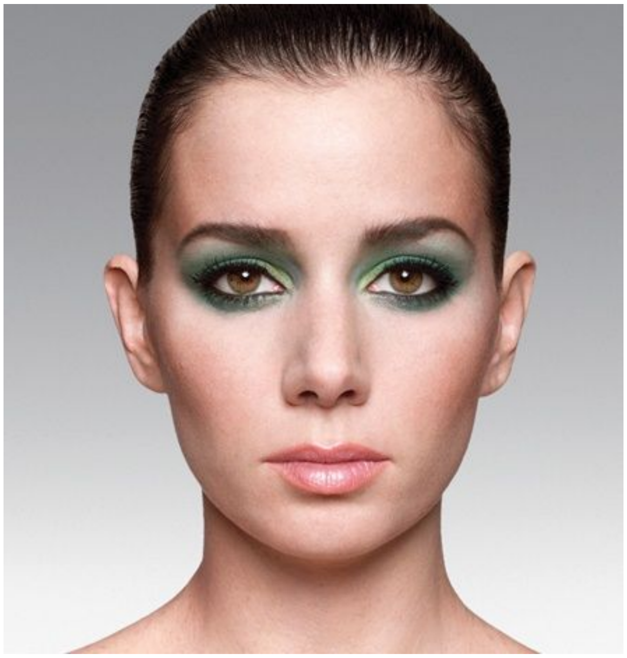
1. Make sure that you select the forehead points appropriately.
2. Lip-transfer algorithm runs 4 loops and can take a huge amount of time (~20 mins). Hence, we are copying pixel value in the resultant image using triangle correspondence.
3. Possible Ghosting Effects in eyebrows, and some cases of partial eye transfer were also found.

# TESTING

SAMPLE IMAGE 2:

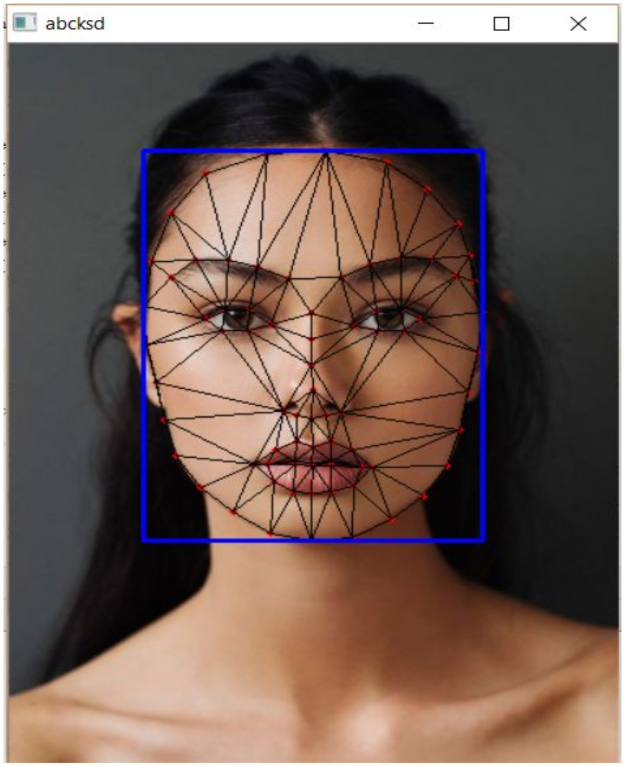


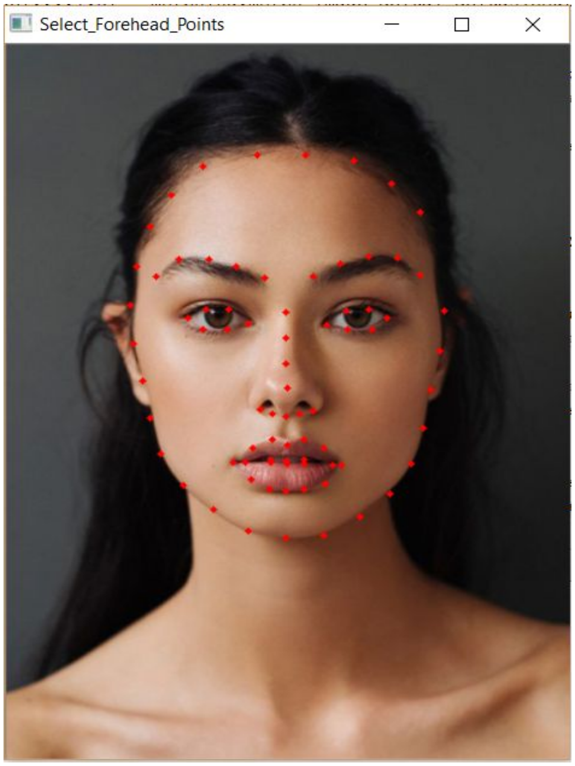
REFERENCE IMAGE 2:

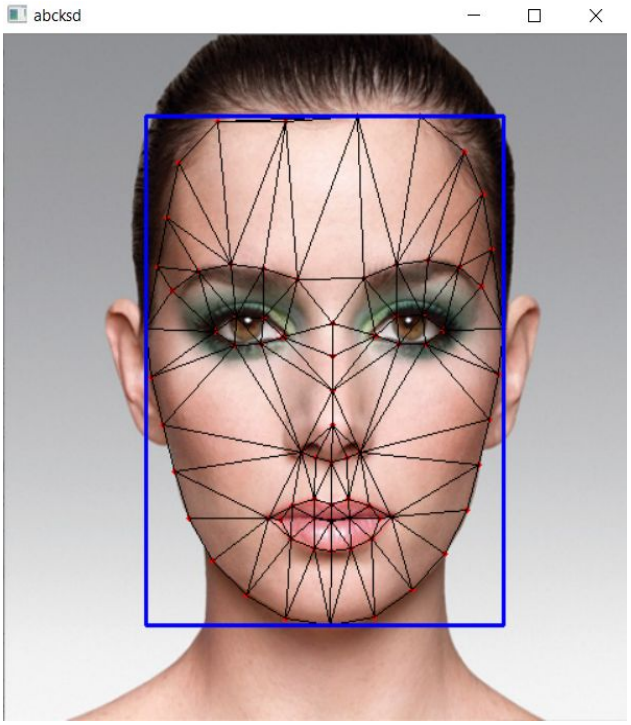


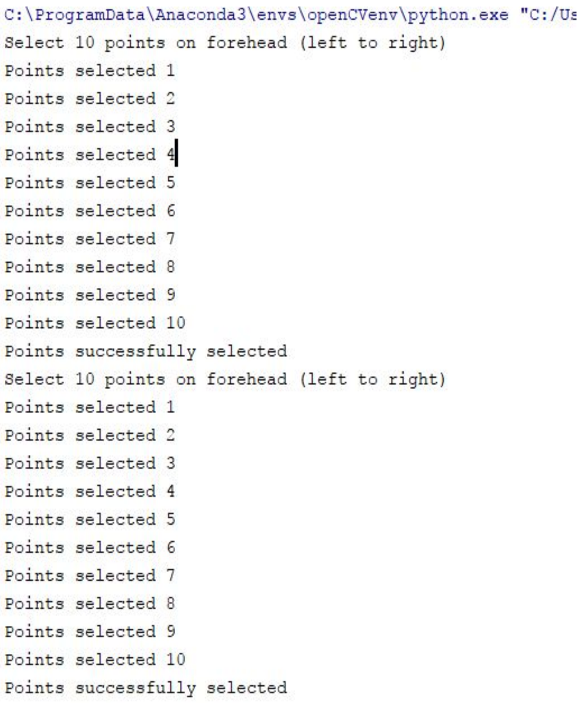
# Image outputs in different pipeline stages:

## STAGE 1: FACE ALIGNMENT



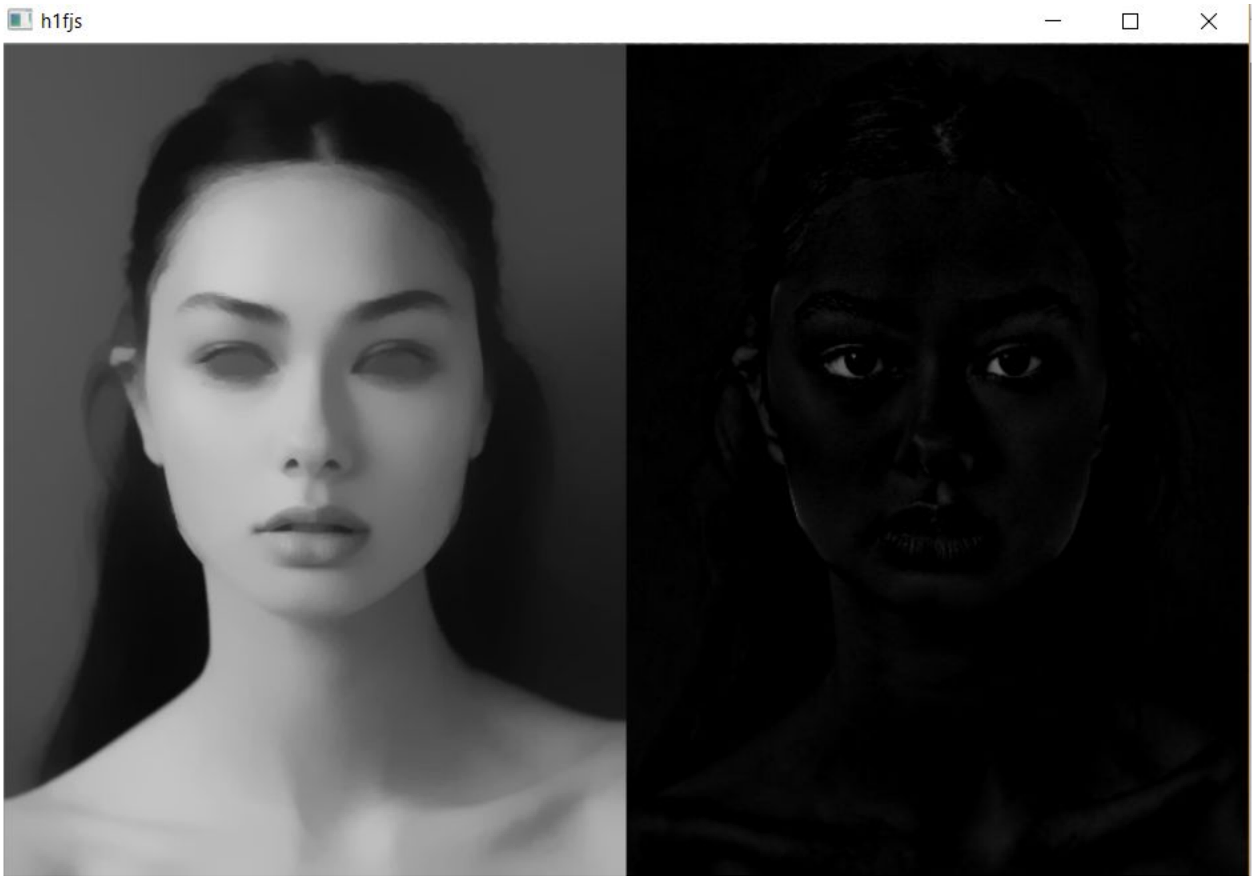


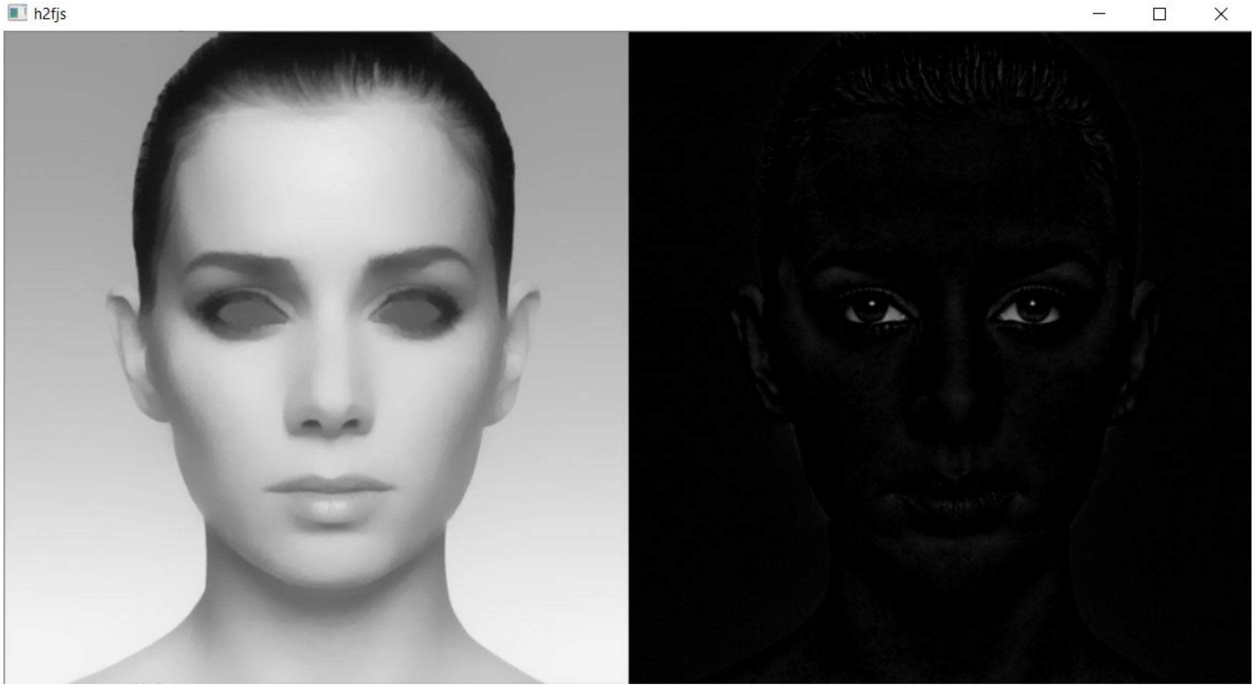




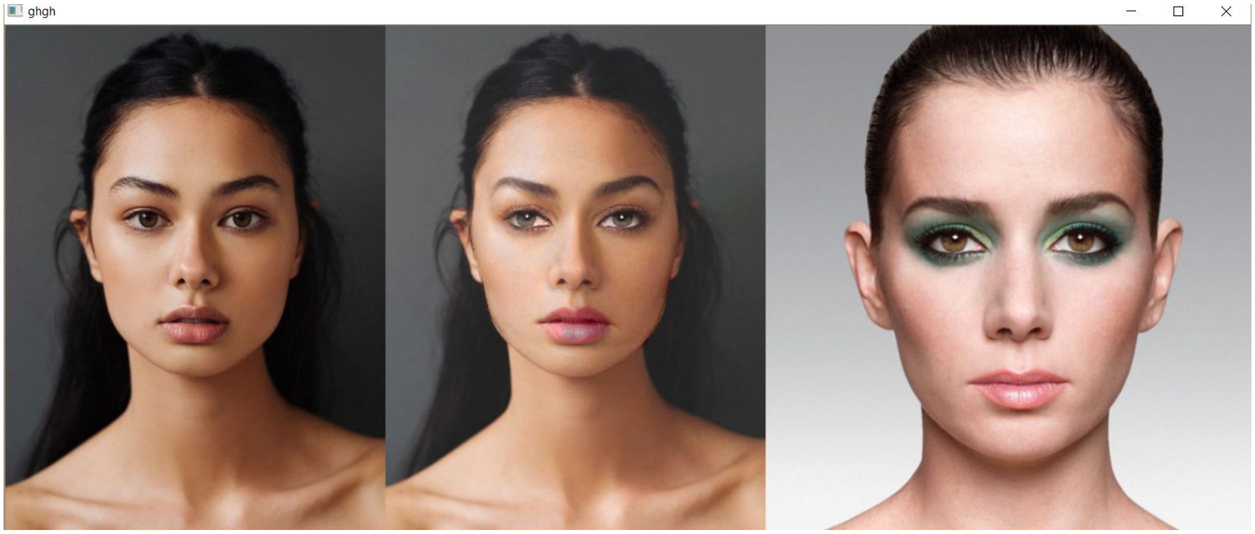
```
C:\ProgramData\Anaconda3\envs\openCVenv\python.exe "C:/Us
Select 10 points on forehead (left to right)
Points selected 1
Points selected 2
Points selected 3
Points selected 4
Points selected 5
Points selected 6
Points selected 7
Points selected 8
Points selected 9
Points selected 10
Points successfully selected
Select 10 points on forehead (left to right)
Points selected 1
Points selected 2
Points selected 3
Points selected 4
Points selected 5
Points selected 6
Points selected 7
Points selected 8
Points selected 9
Points selected 10
Points successfully selected
```

# STAGE 2: LAYER DECOMPOSITION

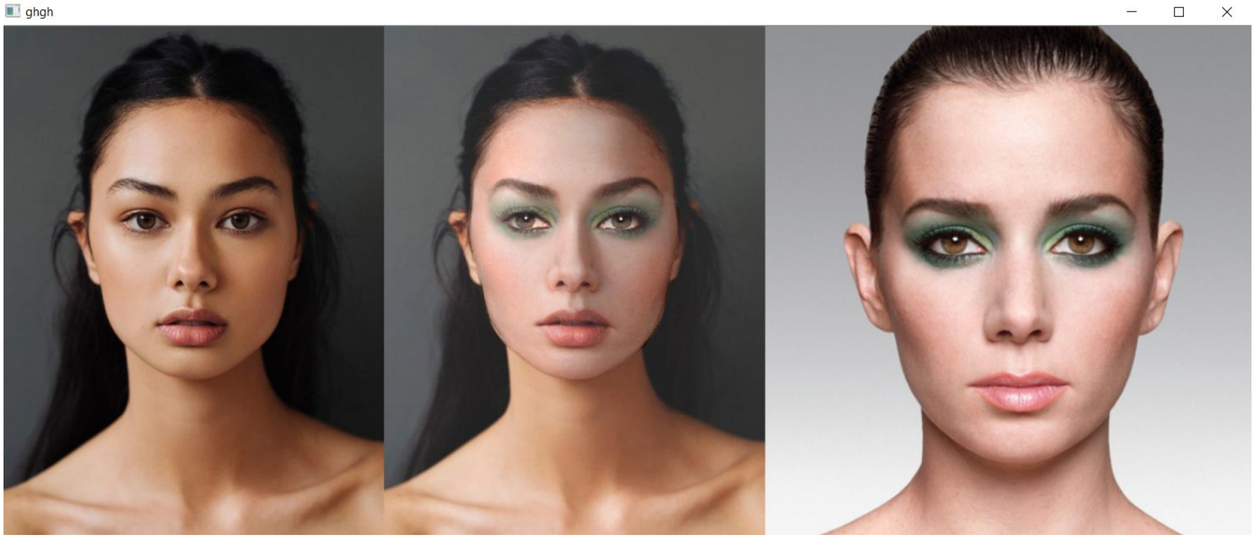




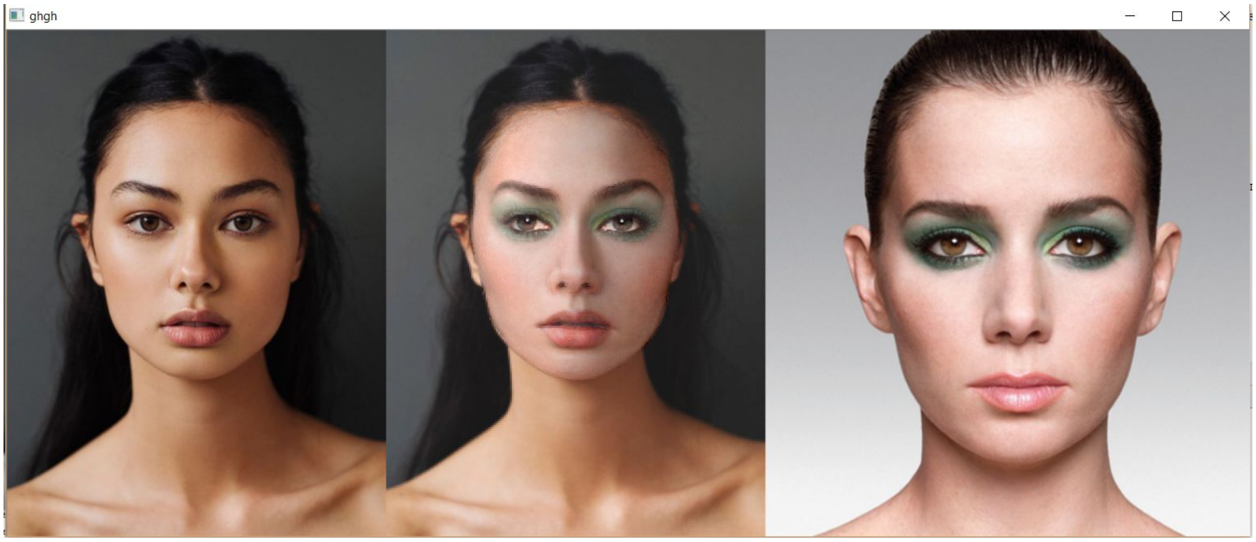
## STAGE 3: SKIN DETAIL TRANSFER



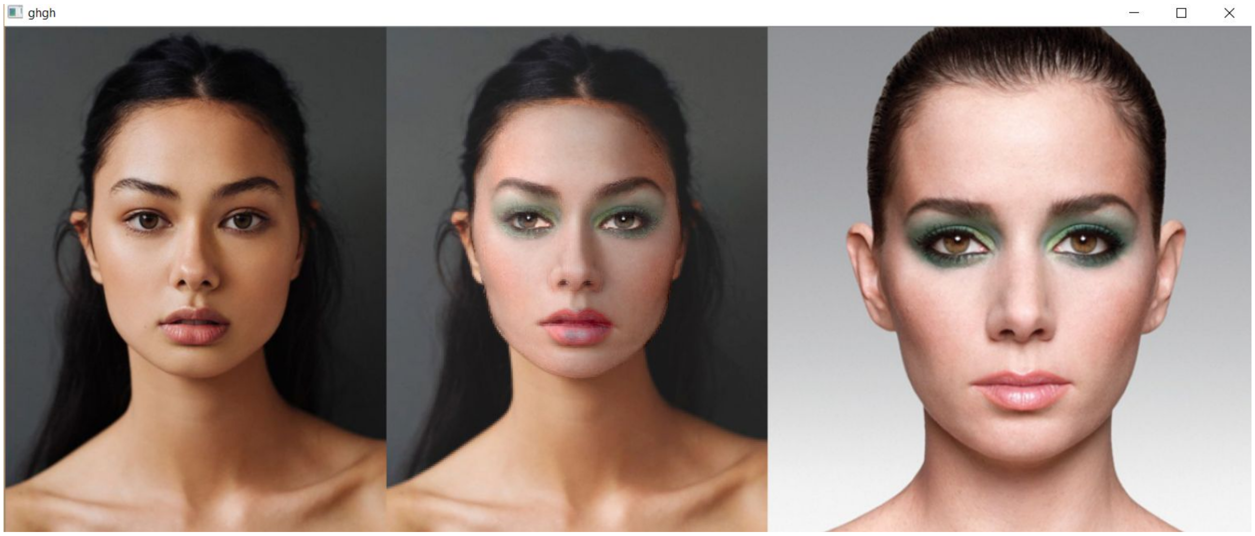
## STAGE 4: COLOR TRANSFER



## STAGE 5: HIGHLIGHT AND SHADING TRANSFER



## STAGE 6: LIP MAKEUP



**Concluding Remark:** Through this assignment, we were able to learn morphing and its implementation in real-life scenarios such as make-up. This assignment made us more comfortable in using opencv environment. It allowed us to separate an image into different layers, and deal with layers individually, and then combine all the things together.