FACULTY OF COMPUTING AND INFORMATICS (FCI)

SUBJECT CODE : TDS 3651
SUBJECT NAME : Visual Information Processing

Assignment 1

TRIMESTER 1, 2021/2022

Prepared by

Student Name : Lee Xi Jie

Student ID : 1161204459

Subject Lecturer,
Dr. Loh Yuen Peng

Table of Contents

1.0 Overview

1.1 Abstract

This report presents a technique for extracting face characteristics from a noisy human picture. There have been numerous attempts to resolve the difficulties, but the fact that students are not permitted to utilise a deep learning technique or any pre-trained model to do segmentations remains a challenge. Instead of utilising third-party libraries, a simple technique has been presented that uses the built-in python cv2, numpy packages to do facial features segmentation. Filtering, sharpening, masking, and morphological procedures make up the framework of this approach. When evaluated with 50 human face pictures, this approach produced a performance-adjusted rand error of 27%, accuracy of 70%, recall of 78%, and IoU of 60%.

1.2 Introduction

Facial features segmentation is a critical topic of mid level vision which aims at jointly categorizing and grouping image regions into coherent parts. Studies by [1], the researchers get a very good performance which is 93% in accuracy. In the real world, an application for skin beautification could certainly benefit from a method which disambiguates ”real” skin from mouth and eyes. For these reasons, facial features can be classified into six classes: skin, hair, eyes, nose, mouth, and background.

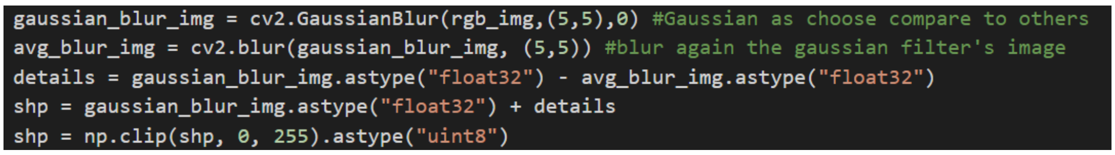
The report's abstract and introduction are described in Chapter 1. Chapter 2 describes the proposed framework, as well as the accompanying visuals and related coding. In addition, the method's findings are presented and analysed in Chapter 3. At the end of Chapter 4, some of the ideas are discussed for future efforts to enhance the algorithms.

2.0 Methodology

This chapter will basically explain the techniques used for the segmentations of each facial feature.

2.1 Image Sharpening

While applying the sharpening process into the image, it will generate a more clear color intensity difference. Besides, the edges from the image will have a greater intensity value which will provide benefits when doing masking technique.



```
gaussian_blur_img = cv2.GaussianBlur(rgb_img,(5,5),0) #Gaussian as choose compare to others
avg_blur_img = cv2.blur(gaussian_blur_img, (5,5)) #blur again the gaussian filter's image
details = gaussian_blur_img.astype("float32") - avg_blur_img.astype("float32")
shp = gaussian_blur_img.astype("float32") + details
shp = np.clip(shp, 0, 255).astype("uint8")
```

Figure 2.1 Implementation of sharpening process

From the figure 2.1 above, the sharpening process is applied into the human face image. Firstly, a smoothing process is applied into the images which can help to remove the noise, and some “high frequency” values such as edges.

For instance, I had to apply gaussian blur with kernel size (5,5) to remove noise from the RGB image. After the first smoothing, I also apply average blur with kernel (5,5) on the gaussian blur image and the next step is to get the details of both blurred images which is subtracting between gaussian blur and average blur. From the studies, I knew that the averaging blur tends to blur the image more as compared to gaussian blur; which the gaussian blur is providing a stronger value at the center when compared with average blur. The reason both are in data type float32 is to avoid overflow issues. After that by adding the details into a gaussian blur image, it will generate a sharpened image with uint8 which will restrict the pixel value between 0 and 255.

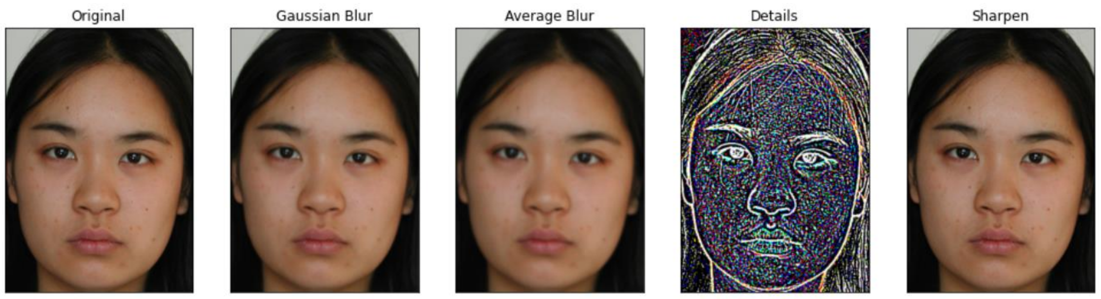
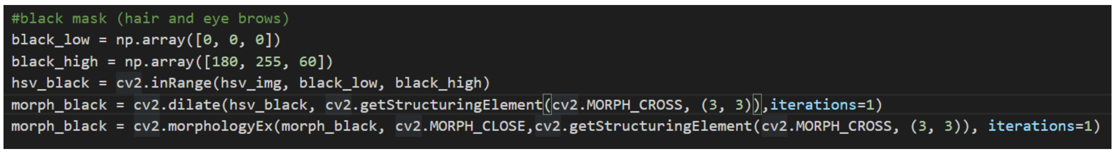


Figure 2.2 Result of sharpening process

The figure 2.2 above shows the results of the sharpening process.

2.2 Masking using HSV

2.2.1 Black Color



```
#black mask (hair and eye brows)
black_low = np.array([0, 0, 0])
black_high = np.array([180, 255, 60])
hsv_black = cv2.inRange(hsv_img, black_low, black_high)
morph_black = cv2.dilate(hsv_black, cv2.getStructuringElement(cv2.MORPH_CROSS, (3, 3)),iterations=1)
morph_black = cv2.morphologyEx(morph_black, cv2.MORPH_CLOSE,cv2.getStructuringElement(cv2.MORPH_CROSS, (3, 3)), iterations=1)
```

Figure 2.3 Implementation of black hsv mask

The figure 2.3 shows the implementation code for masking black color in hsv color space. I had applied a black colour mask on a hsv image with threshold [0, 0, 0] to [180, 255, 60]. After setting the threshold, I can generate a black mask based on the hsv image to obtain a binary image. Not only masking the black color out, I had also applied the morphological dilate and closing to it. The reason to apply dilate and closing to the mask is to increase the connection between the intensity as a sample result shown in figure 2.4 below.

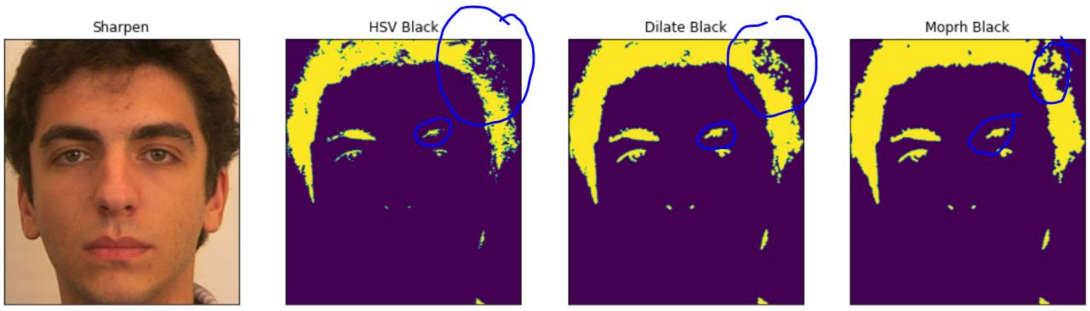
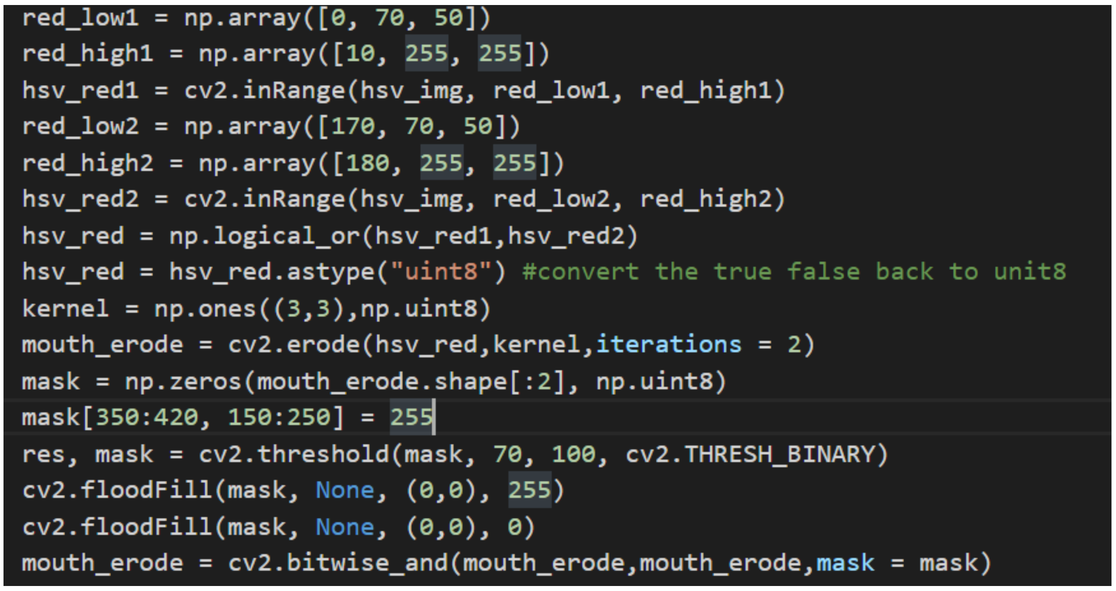


Figure 2.4 Result of Black HSV Mask

2.2.2 Red Color



```
red_low1 = np.array([0, 70, 50])
red_high1 = np.array([10, 255, 255])
hsv_red1 = cv2.inRange(hsv_img, red_low1, red_high1)
red_low2 = np.array([170, 70, 50])
red_high2 = np.array([180, 255, 255])
hsv_red2 = cv2.inRange(hsv_img, red_low2, red_high2)
hsv_red = np.logical_or(hsv_red1,hsv_red2)
hsv_red = hsv_red.astype("uint8") #convert the true false back to unit8
kernel = np.ones((3,3),np.uint8)
mouth_erode = cv2.erode(hsv_red,kernel,iterations = 2)
mask = np.zeros(mouth_erode.shape[:2], np.uint8)
mask[350:420, 150:250] = 255
res, mask = cv2.threshold(mask, 70, 100, cv2.THRESH_BINARY)
cv2.floodFill(mask, None, (0,0), 255)
cv2.floodFill(mask, None, (0,0), 0)
mouth_erode = cv2.bitwise_and(mouth_erode,mouth_erode,mask = mask)
```

Figure 2.5 Implementation of red hsv mask (mouth)

The figure 2.5 shows the implementation code for HSV Red Mask. I had applied a red colour mask on an HSV image with 4 thresholds [0, 70, 50] to [10, 255, 255] and [170,70,50] to [180,255,255] . After setting the thresholds, I can generate a red mask based on the HSV image to obtain a binary image. Not only masking the red color out, I had also applied the morphological erode and a location masking to detect the mouth. The reason to apply erode and location masking to the mask is to get the mouth out because and ignore the skin figure 2.6 below.

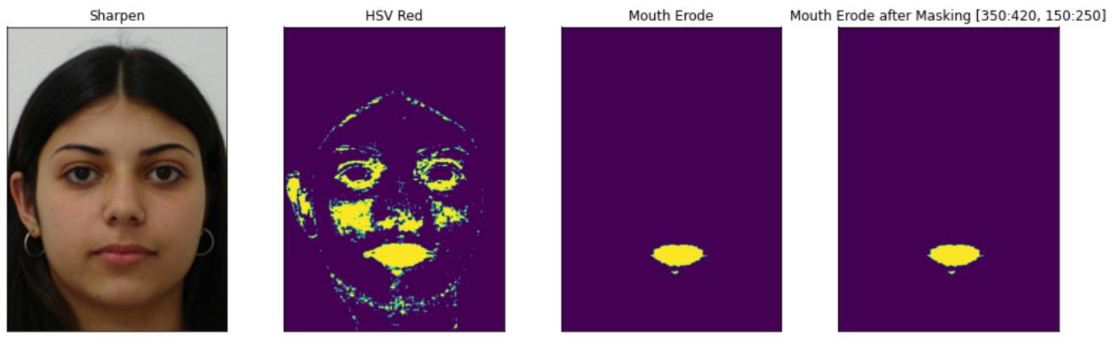
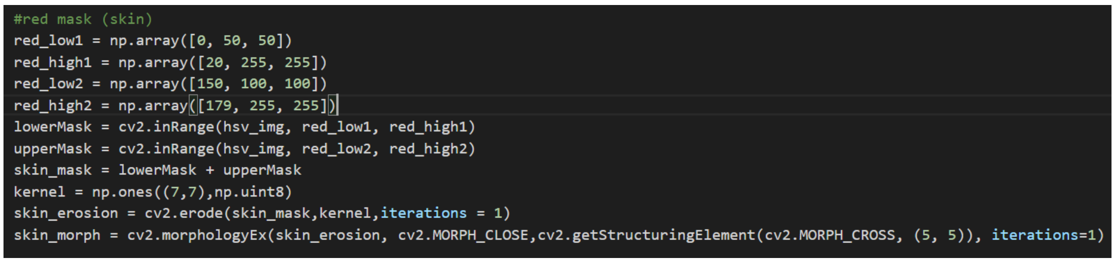


Figure 2.6 Result of red hsv mask (mouth)

Besides, the skin is also considered red in the dataset after studying the datasets given.



```
#red mask (skin)
red_low1 = np.array([0, 50, 50])
red_high1 = np.array([20, 255, 255])
red_low2 = np.array([150, 100, 100])
red_high2 = np.array([179, 255, 255])
lowerMask = cv2.inRange(hsv_img, red_low1, red_high1)
upperMask = cv2.inRange(hsv_img, red_low2, red_high2)
skin_mask = lowerMask + upperMask
kernel = np.ones((7,7),np.uint8)
skin_erosion = cv2.erode(skin_mask,kernel,iterations = 1)
skin_morph = cv2.morphologyEx(skin_erosion, cv2.MORPH_CLOSE,cv2.getStructuringElement(cv2.MORPH_CROSS, (5, 5)), iterations=1)
```

Figure 2.7 Implementation of red hsv mask (skin)

The figure 2.7 shows the implementation code for red hsv mask. I had applied a red colour mask on a hsv image with 4 thresholds [0, 50, 50] to [20, 255, 255] and [150,100,100] to [179,255,255] . After setting the thresholds, I can generate a red mask from the skin based on the hsv image to obtain a binary image. Besides morphological operation is also applied into this mask to erode and close to the images. There are some reasons for applying morphology to the images because some of the hair of the human is considered red after erode and closing. The difference can be seen clearly as result shown in figure 2.8 below.

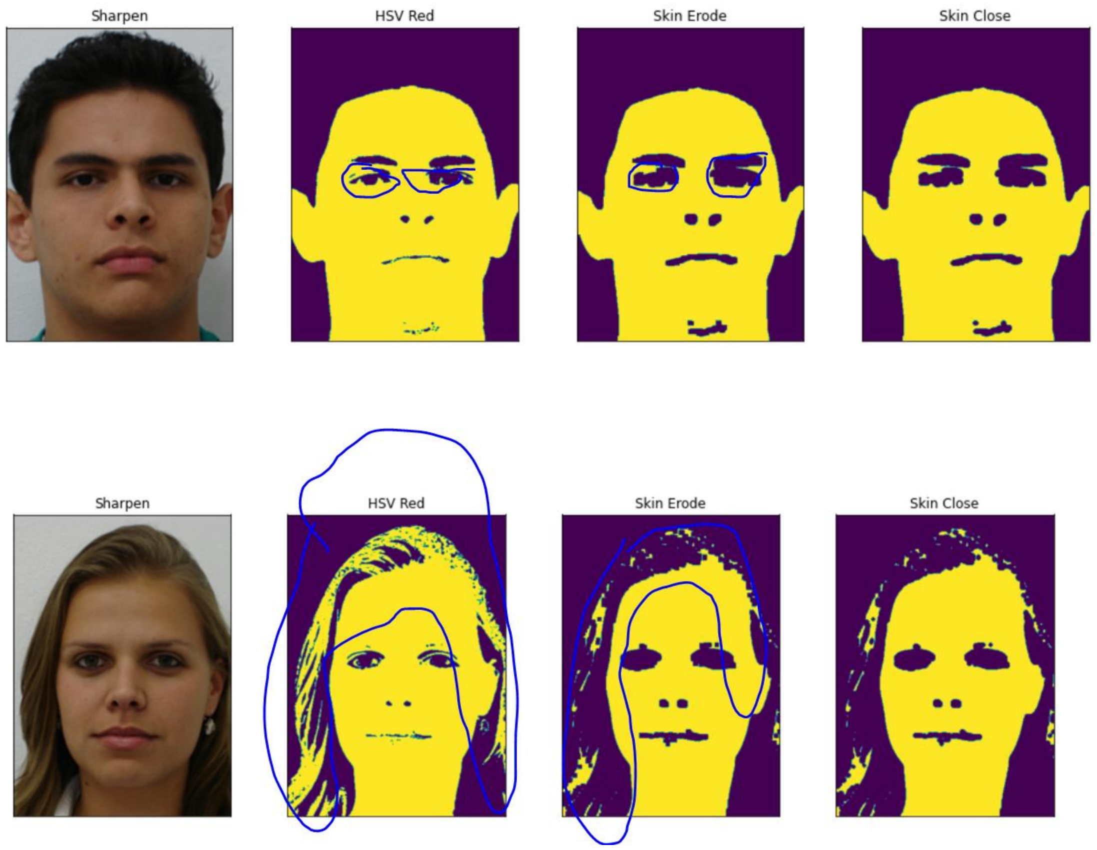
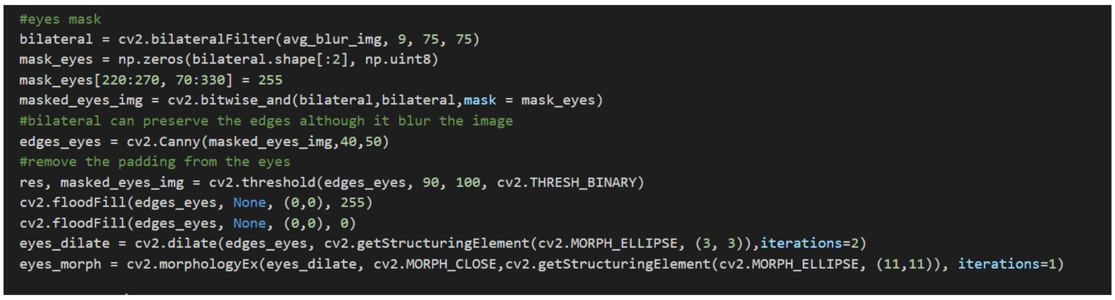


Figure 2.8 Result of Red HSV Mask (Skin)

2.3 Masking Using Edge Detection and Location

2.3.1 Masking Eyes



```
#eyes mask
bilateral = cv2.bilateralFilter(avg_blur_img, 9, 75, 75)
mask_eyes = np.zeros(bilateral.shape[:2], np.uint8)
mask_eyes[220:270, 70:330] = 255
masked_eyes_img = cv2.bitwise_and(bilateral,bilateral,mask = mask_eyes)
#bilateral can preserve the edges although it blur the image
edges_eyes = cv2.Canny(masked_eyes_img,40,50)
#remove the padding from the eyes
res, masked_eyes_img = cv2.threshold(edges_eyes, 90, 100, cv2.THRESH_BINARY)
cv2.floodFill(edges_eyes, None, (0,0), 255)
cv2.floodFill(edges_eyes, None, (0,0), 0)
eyes_dilate = cv2.dilate(edges_eyes, cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (3, 3)),iterations=2)
eyes_morph = cv2.morphologyEx(eyes_dilate, cv2.MORPH_CLOSE,cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (11,11)), iterations=1)
```

Figure 2.9 Implementation of edge detection (eyes)

While segmenting eyes, I have applied an edge detection to mask it. The figure 2.9 shows the implementation of coding. In the first step, I have applied a bilateral filter first before doing edge detection. From the research, I knew that although bilateral will blur the image, it can perverse the edges in the image. Besides, a location masking also applies to this approach in which I locate the most of the human eye's location which is [220:270, 70:330]. After masking out the location, I use the canny edge detection with threshold [40,50] and apply the morphological operations such as dilate and closing to the eyes segmentations.

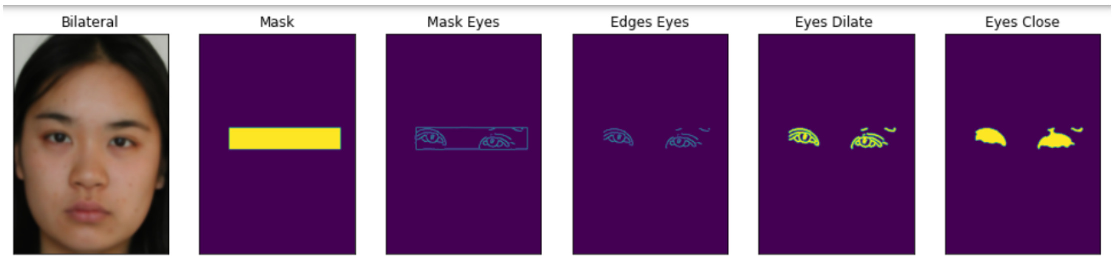
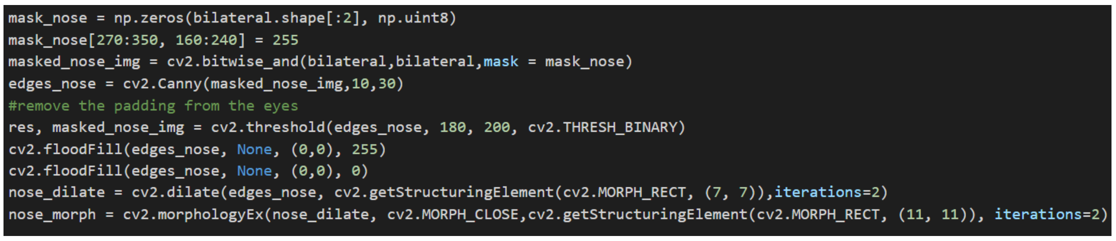


Figure 2.10 Result of edge detection (eyes)

Figure 2.10 shows the result of using canny edge detection approach which is filtering with bilateral, masking the location and applying morphological operations at the end.

2.3.2 Masking Nose



```
mask_nose = np.zeros(bilateral.shape[:2], np.uint8)
mask_nose[270:350, 160:240] = 255
masked_nose_img = cv2.bitwise_and(bilateral,bilateral,mask = mask_nose)
edges_nose = cv2.Canny(masked_nose_img,10,30)
#remove the padding from the eyes
res, masked_nose_img = cv2.threshold(edges_nose, 180, 200, cv2.THRESH_BINARY)
cv2.floodFill(edges_nose, None, (0,0), 255)
cv2.floodFill(edges_nose, None, (0,0), 0)
nose_dilate = cv2.dilate(edges_nose, cv2.getStructuringElement(cv2.MORPH_RECT, (7, 7)),iterations=2)
nose_morph = cv2.morphologyEx(nose_dilate, cv2.MORPH_CLOSE,cv2.getStructuringElement(cv2.MORPH_RECT, (11, 11)), iterations=2)
```

Figure 2.11 Implementation of edge detection (nose)

Segmenting the nose is the most challenging in this Assignment. The method I used to segment the nose is masking a location and doing an edge detection. The figure 2.11 shows the implementation of coding. In the first step, I have applied a bilateral filter first before doing edge detection. Besides, a location masking also applies to this approach in which I locate the most of the human nose’s location which is [270:350, 160:240]. After masking out the location, I use the canny edge detection with threshold [100,200] and apply the morphological operations such as dilate and closing to the nose segmentations.

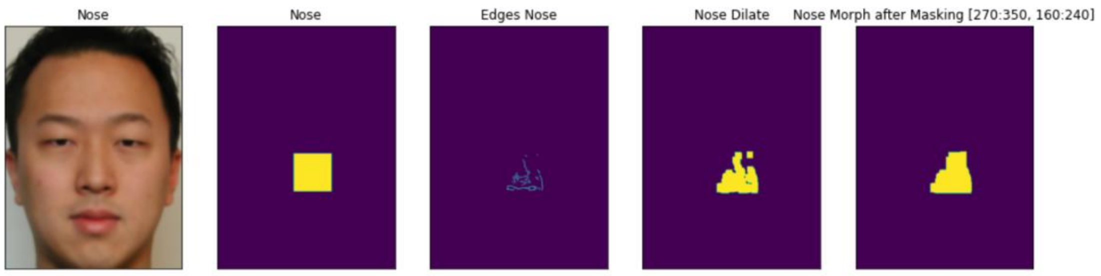
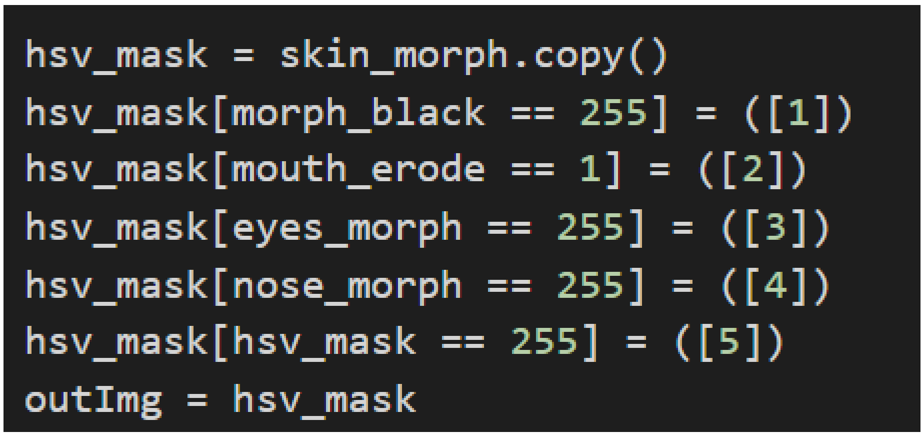


Figure 2.12 Result of edge detection (nose)

Figure 2.12 shows the result of using canny edge detection approach which filters with bilateral, masking the location and applying morphological operations at the end.

2.4 Merging all the masking



```
hsv_mask = skin_morph.copy()
hsv_mask[morph_black == 255] = ([1])
hsv_mask[mouth_erode == 1] = ([2])
hsv_mask[eyes_morph == 255] = ([3])
hsv_mask[nose_morph == 255] = ([4])
hsv_mask[hsv_mask == 255] = ([5])
outImg = hsv_mask
```

Figure 2.13 Implementation of merging all the segments

After segmenting all the features, I have to merge all of them into a single image. Figure 2.13 shows the implementation of coding to merge all the features.

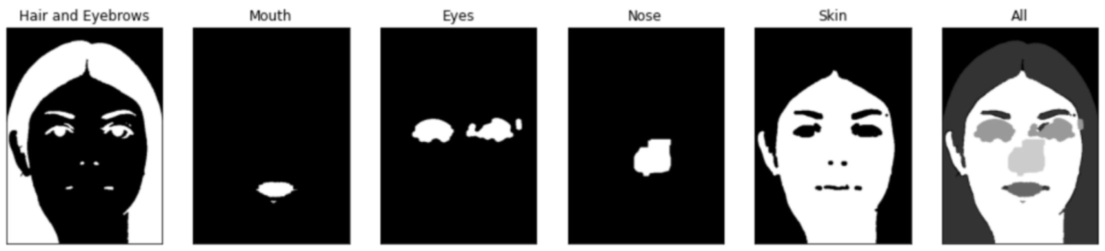


Figure 2.14 Results of merging all the segmentations

Figure 2.14 shows the final output after merging all the features from a human face. The further discuss of the result and the analysis will be discuss in the chapter 3.

3.0 Result and Analysis

This chapter will explain more details and analysis of the output results.

3.1.1 Result

Adapted Rand Error	27%
Precision	70%
Recall	78%
IoU	60%

Table 3.1 Results

3.1 Color Masking

3.1.1 Black Color Masking

The figure below shows some of the images output to be further discussed and analysed.

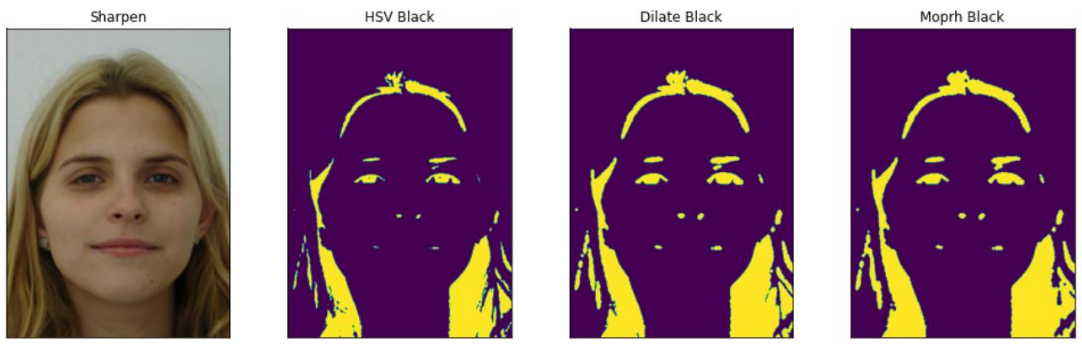


Figure 3.1 Black hsv issue 1

Figure 3.1 above shows the issues when masking human hair. For instance, in this human, the hair color is considered yellow, hence when I apply black color it will not extract completely what we humans expect.

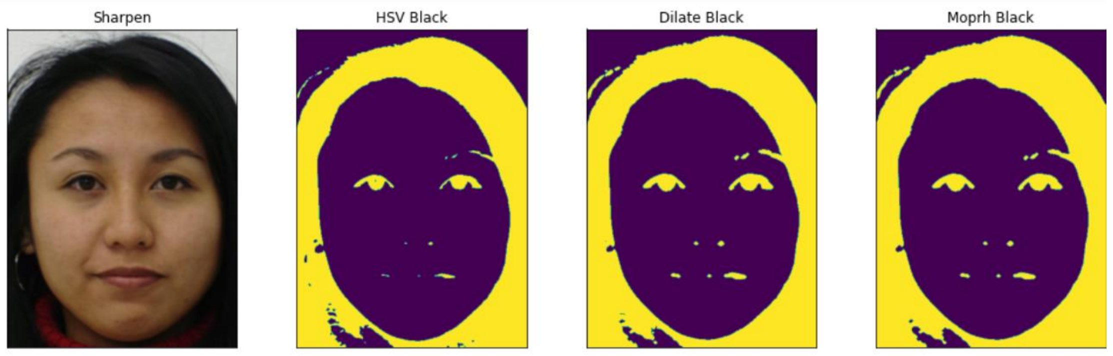


Figure 3.2 Black hsv issue 2

Besides, there is also a shadow issue occurring when I do black color masking as sample result shown as figure 3.2. Other than the shadow issues, some of the eye brows will be eliminated as they are not in the range of the threshold set.

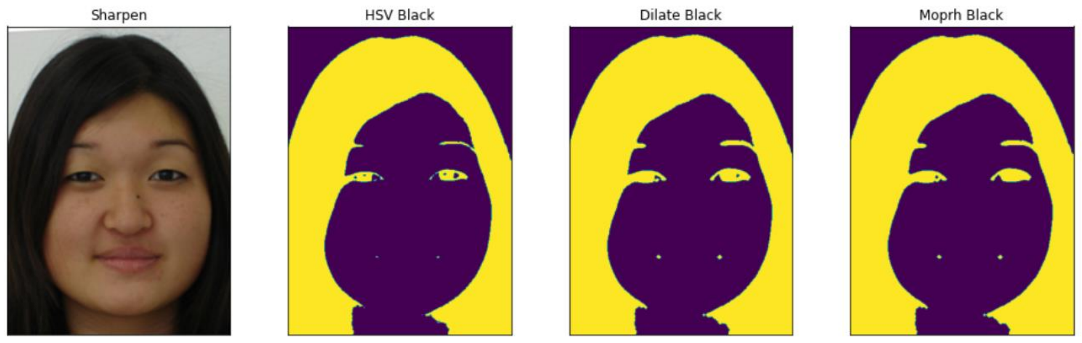


Figure 3.3 Black color masking

Meanwhile there are also some of the results that are well performed which extract what I wish. For example, in the figure 3.3 above, it will exactly segment what I expect for the hair. The eyes will be replaced when I do the masking for the eyes feature.

3.1.2 Red Color Masking

The figure below shows some of the images output to be further discussed and analysed.

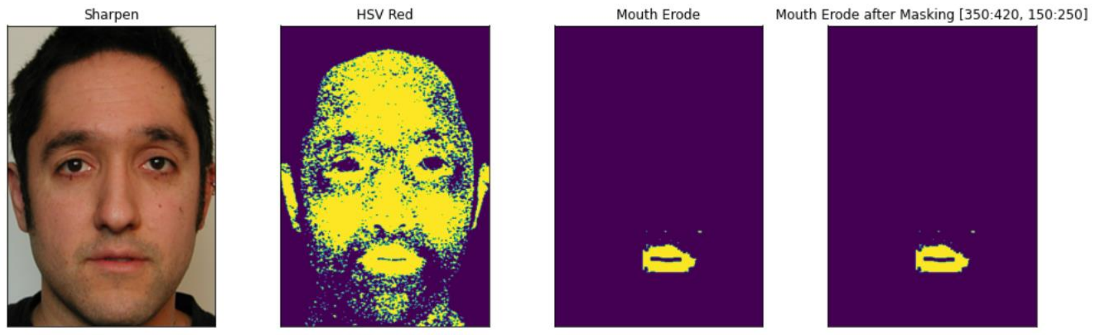


Figure 3.4 Red hsv (mouth)

From the figure 3.4 above shows some of the common issues that are found in the visualization. For example, I can clearly see the human skin is slightly near the red color range. Hence, the location masking is applied here. For example, I have manually input the mouth location and masking it out.

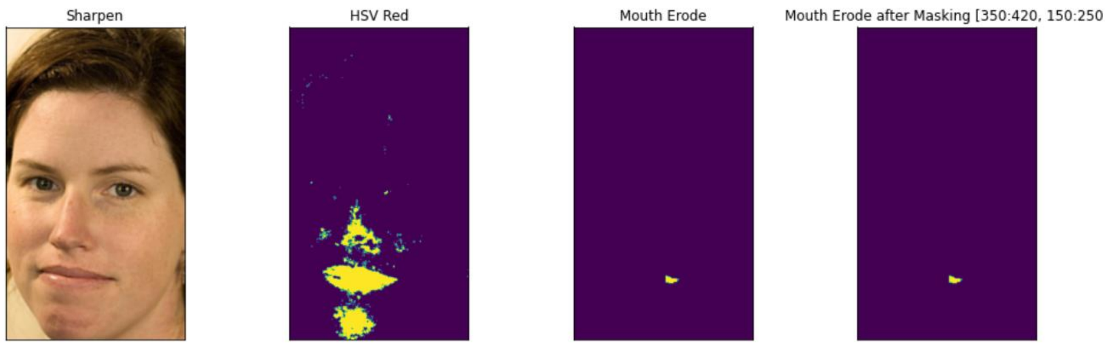


Figure 3.5 Red hsv mouth issue 1

Even some of the images can be well performed but there are also issues occurring when doing the location masking. For this image, the human’s mouth is slightly to the left, as the location I input was center, so the machine will eliminate the mouth. After studying the input data, this method is still applied because most of the human mouth is in the center of the bottom part.

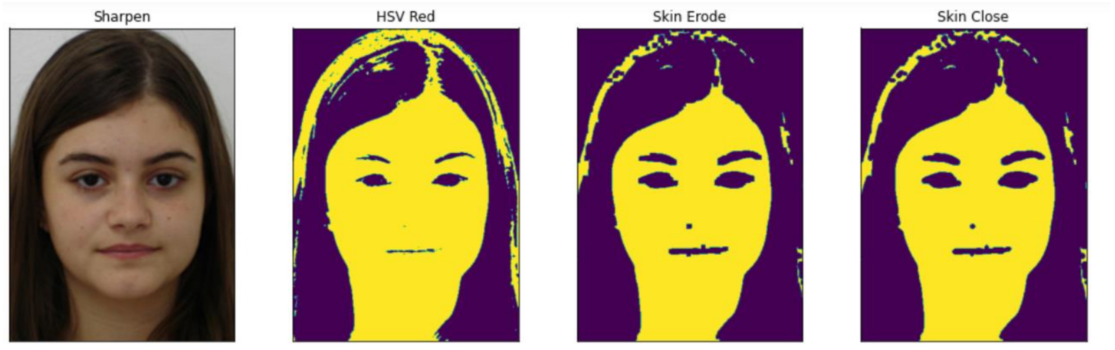


Figure 3.6 Red hsv (skin)

Besides, after studying the dataset, I knew that red color is also suitable for masking skin segmentation. There is only least image are not suitable as the background will conflict with the skin.

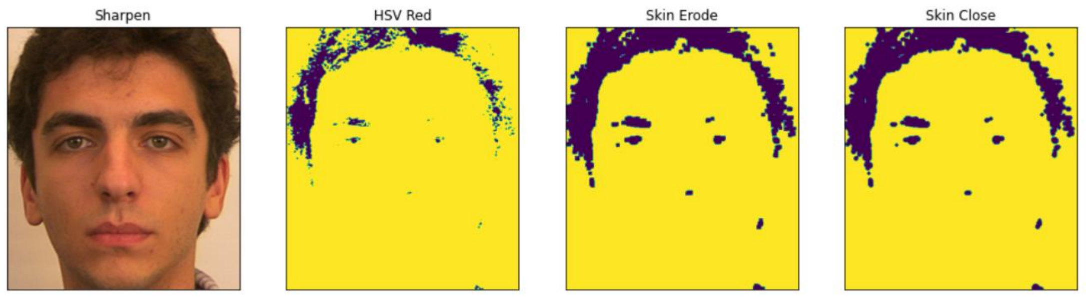


Figure 3.7 Red hsv skin issue 1

From the figure 3.7, the issue occurs of conflict color with background. Even though I have set the multiple threshold to eliminate this issue, the background still cannot be eliminated in my proposed algorithms.

3.2 Edge Detection

3.2.1 Edge Detection from Eyes

The figure below shows some of the images output to be further discussed and analysed.

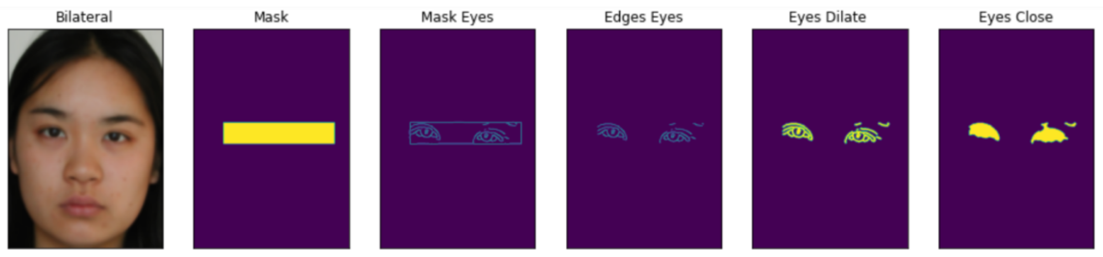


Figure 3.8 Location masking and edge detection

From the figure 3.8, it shows a good result in segmenting the human’s eyes using edge detection with location. Theoretically this method considers hard coding to the studied dataset. If the eye is not exactly in the location, it will not be able to be segmented out.

3.2.2 Edge Detection in Nose

Nose edge detection is the most challenging in this assignment. As if I only HSV color to segment out, it will conflict with skin. Hence, I also have edge detection in the human's nose. The figure below shows some of the images output to be further discussed and analysed.

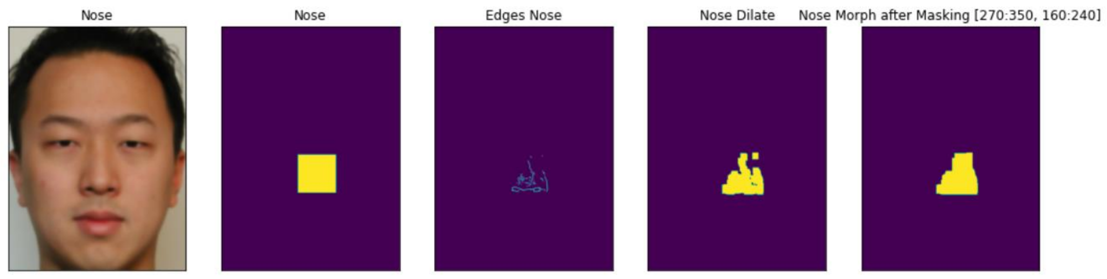


Figure 3.9 Location masking and edge detection (nose)

From the figure 3.9, I mask the center of the image which is the nose, after that the edges and nose are detected and apply morphological operation to get the final result.

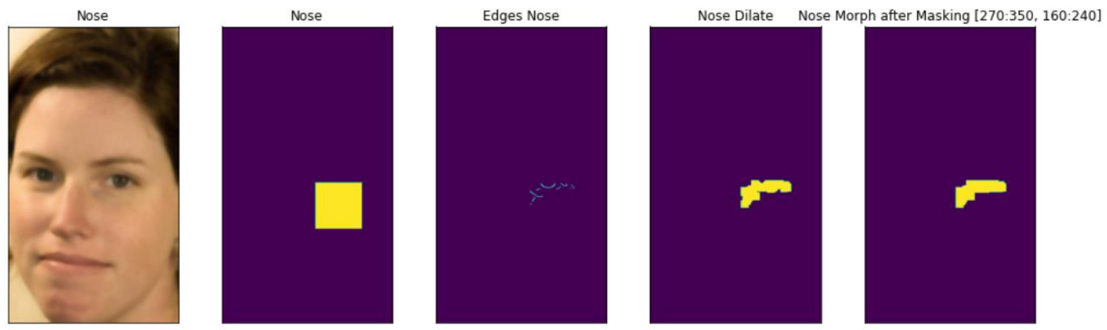


Figure 3.10 Location masking and edge detection (nose) issue 1

For example in the figure 3.10, it will compute a wrong result as the nose is not in the center of the image.

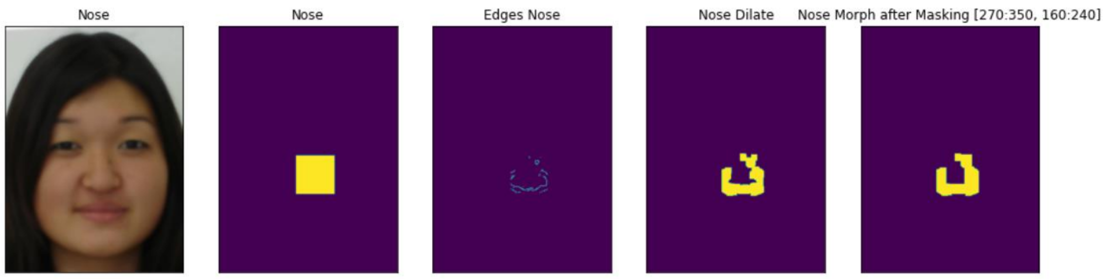


Figure 3.11 Location masking and edge detection (nose) issue 2

Figure 3.11 shows the result of the human's nose. After applying dilate and closing, it still could not fulfil what I expected in the beginning.

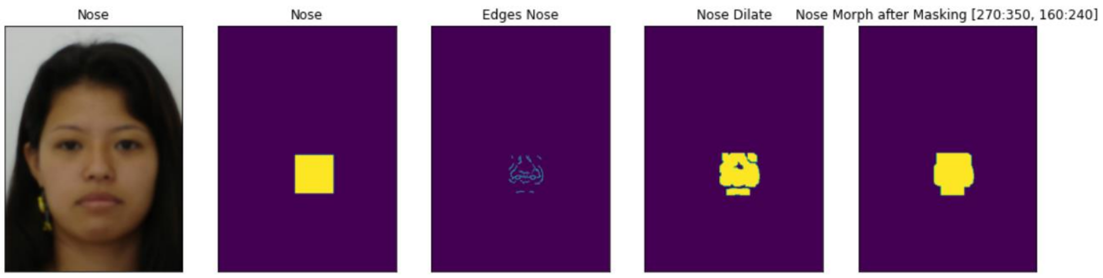
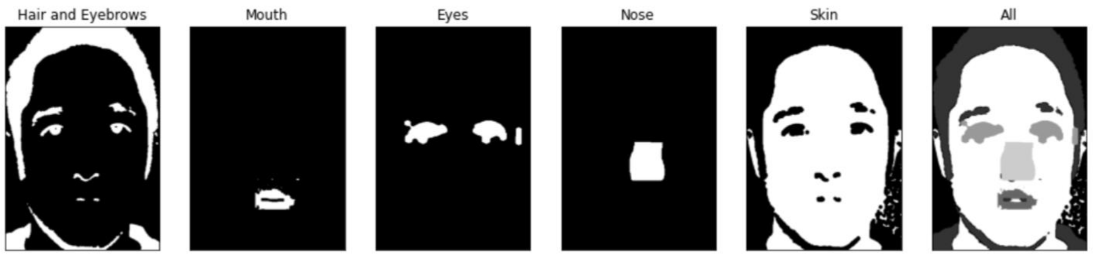


Figure 3.12 Location masking and edge detection (nose) issue 3

As I explain in the chapter 2, I have applied a square masking into the center of the image, so if I dilate some of the noses it will become a square due to the square shape, as shown in figure 3.12.

3.3 Summary Result



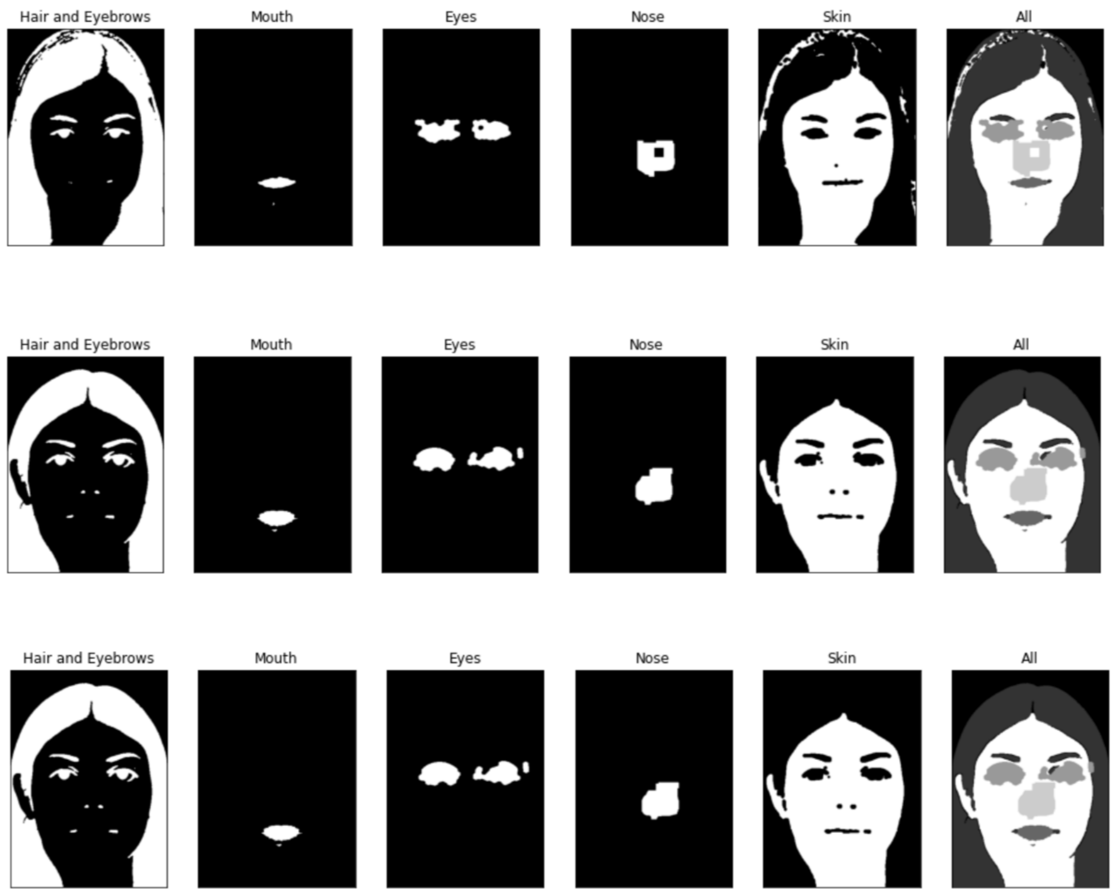
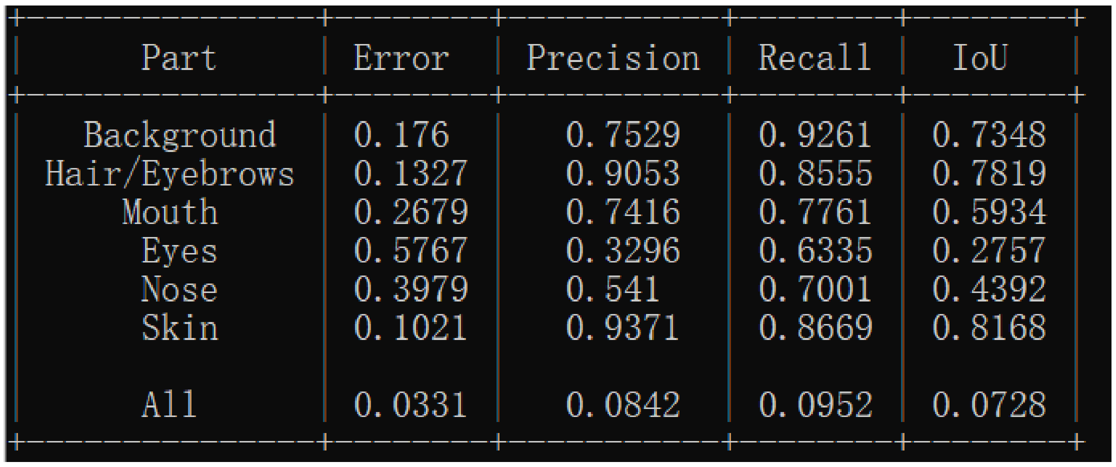


Figure 3.13 Segmentation of facial features

Figure 3.13 shows some of the well performed analysis in this assignment. Besides, the table 3.2 below shows the result for each of the facial features.



```
+----------------+---------+------------+---------+---------+
|      Part      |  Error  | Precision  | Recall  |   IoU   |
+----------------+---------+------------+---------+---------+
|   Background   | 0.176   |   0.7529   | 0.9261  | 0.7348  |
| Hair/Eyebrows  | 0.1327  |   0.9053   | 0.8555  | 0.7819  |
|     Mouth      | 0.2679  |   0.7416   | 0.7761  | 0.5934  |
|      Eyes      | 0.5767  |   0.3296   | 0.6335  | 0.2757  |
|      Nose      | 0.3979  |   0.541    | 0.7001  | 0.4392  |
|      Skin      | 0.1021  |   0.9371   | 0.8669  | 0.8168  |
|                |         |            |         |         |
|      All       | 0.0331  |   0.0842   | 0.0952  | 0.0728  |
+----------------+---------+------------+---------+---------+
```

Table 3.2 Result of the facial features

Overall in the summary, using the hair color hsv mask can generate a well performed result for the hair and eyebrows. Besides, red color is used to segment out the skin and is also getting a 0.9371 which is considered good while comparing to groundtruth. Besides, the algorithms also get an average high result in segmenting background, mouth. To recall while segmenting the mouth, the algorithm used is using red color hsv and locate the bottom part center. Meanwhile, eyes and nose generate an average low result which is only 0.3296 and 0.541 respectively. There are a lot of enhancements that can be reached in this assignment and the details will be further discussed in chapter 4.

4.0 Suggestion for Improvement

In this report, a naïve facial features segmentation is proposed which is including the steps of image smoothing, image sharpening, masking, morphological operation. While applying those techniques, the time it takes is very huge, so I will need to compare the output result with the groundtruth and fine-tuning the parameter until getting a good accuracy. After the second dataset is provided, the performance of the algorithm still consider inaccurate. There is always a better technique to be applied after learning more about the image processing knowledges such as textures. I can get use of texture method to help us to do clustering, superpixels to solve the eyes and nose issues. After studying for some of the online papers, most of the researchers tend to used a deep learning technique to performance a outstanding segmentation.

5.0 Collaborations

In order to complete this assignment, I had some discussion between the classmates. The students listed below have given me guidance while completing this assignment.

Student Name	Student ID
Oi Zhen Fan	1181300513
Lee Wang Lin	1171200748
Cheok Jia Heng	1171201466

6.0 Reference

[1] Khan, Khalil & Mauro, Massimo & Leonardi, Riccardo. (2015). Multi-class semantic segmentation of faces. 10.1109/ICIP.2015.7350915.

[2]https://medium.com/globant/maneuvering-color-mask-into-object-detection-fce61bf891d1