MULTIMEDIA UNIVERSITY

TDS3651
VISUAL INFORMATION PROCESSING

TRIMESTER 1, 2021/2022

ASSIGNMENT

Prepared by

Tee Wai Bing (1171103537)

Abstract

In this assignment, a segmentation algorithm will be created to segment a human face image into 6 parts. There are thresholding methods in HSV image to segment the image's background, nose, skin, hair and eyebrows as well as GrabCut Algorithm for segmenting eyes and mouth.

1.0 Introduction

Image segmentation is a classic problem in computer vision research and has become a hot spot in the field of image understanding. Image segmentation is the first step in image analysis, the foundation of computer vision, and an important part of image understanding, as well as one of the most difficult problems in image processing. Image segmentation is the technique and process of dividing an image into a number of specific areas with unique properties and proposing objects of interest. The objective of this assignment is to design an algorithm that automatically segments a human face image into 6 parts which include the background part, hair and eyebrows part and mouth part, eyes part, nose part and skin part . A set of 50 face images, along with the corresponding ground truth, are provided for me to design the segmentation algorithm.

2.0 Description of Methods

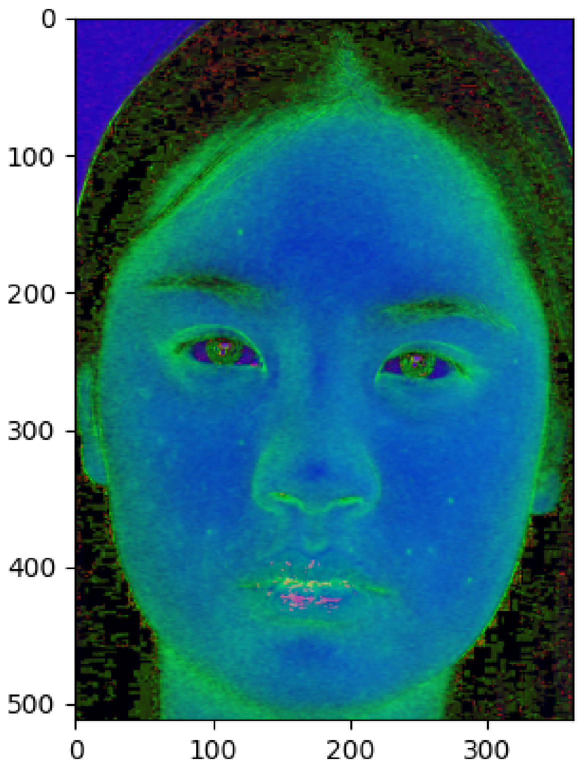
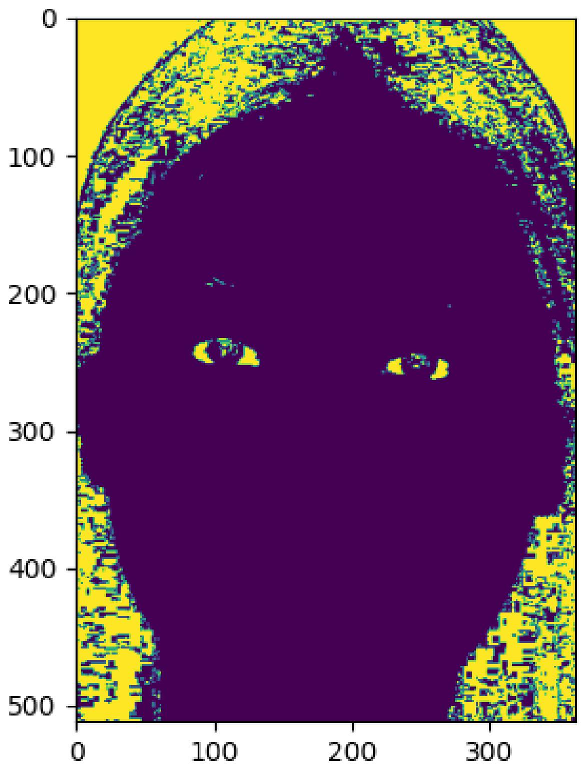
Firstly, I convert the images into the HSV colorspace.

```
img_hsv=cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
```

2.1 Background

I found the background part by using the thresholding in the HSV image. I set a suitable range for the background color in HSV coordinates. Next, I threshold the HSV image using the background color range specified.

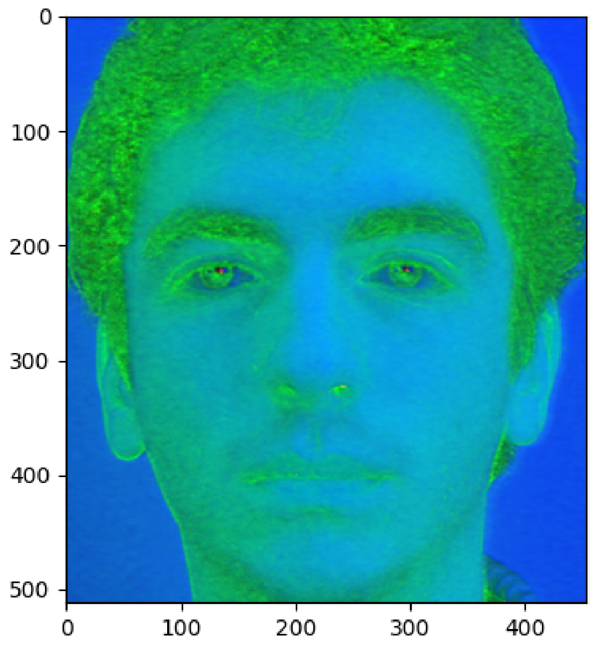
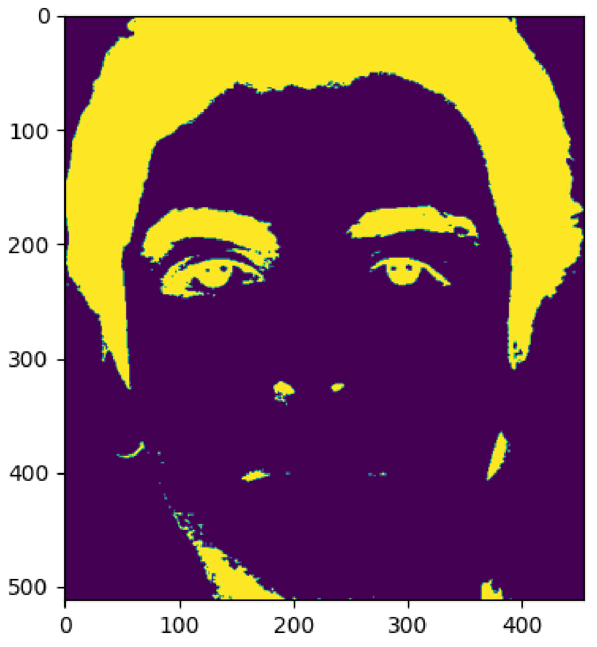
```
lower_bcg = np.array([0,0,0],np.uint8)

upper_bcg = np.array([179,40,255],np.uint8)

bcg = cv2.inRange(img_hsv, lower_bcg, upper_bcg)
```

	
Before Thresholding	After Thresholding

2.2 Hair/Eyebrows

I found the hair and eyebrows part by using the thresholding in the HSV image. I set a suitable range for the hair and eyebrows color in HSV coordinates. Next, I threshold the HSV image using the background color range specified.

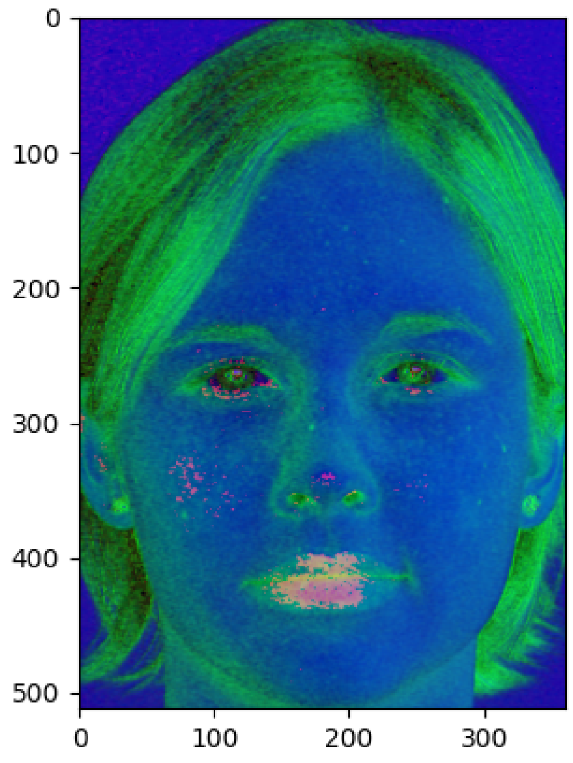
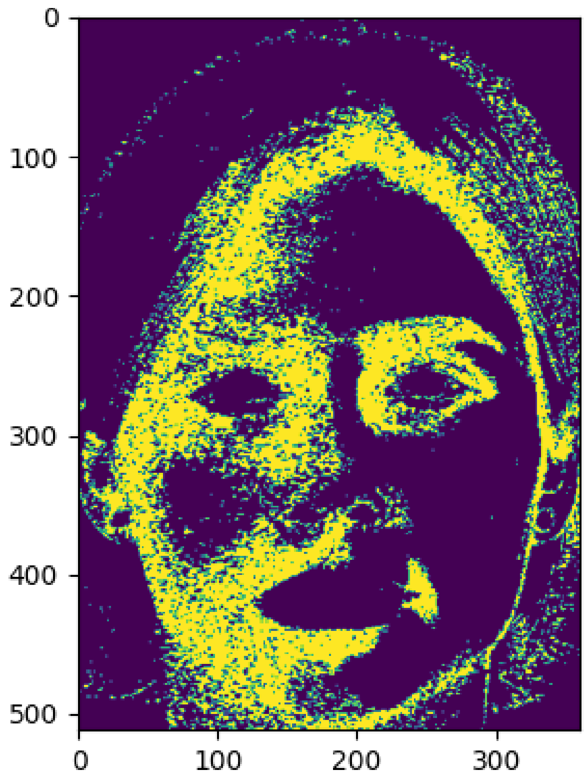
```
lower_hair= np.array([0,0,0],np.uint8)

upper_hair = np.array([179,255,103],np.uint8)

hair = cv2.inRange(img_hsv, lower_hair, upper_hair)
```

 0 100 200 300 400 500 0 100 200 300 400	 0 100 200 300 400 500 0 100 200 300 400
Before Thresholding	After Thresholding

2.3 Nose

I found the nose part by using the thresholding in the HSV image. I set a suitable range for the nose color in HSV coordinates. Next, I threshold the HSV image using the background color range specified.

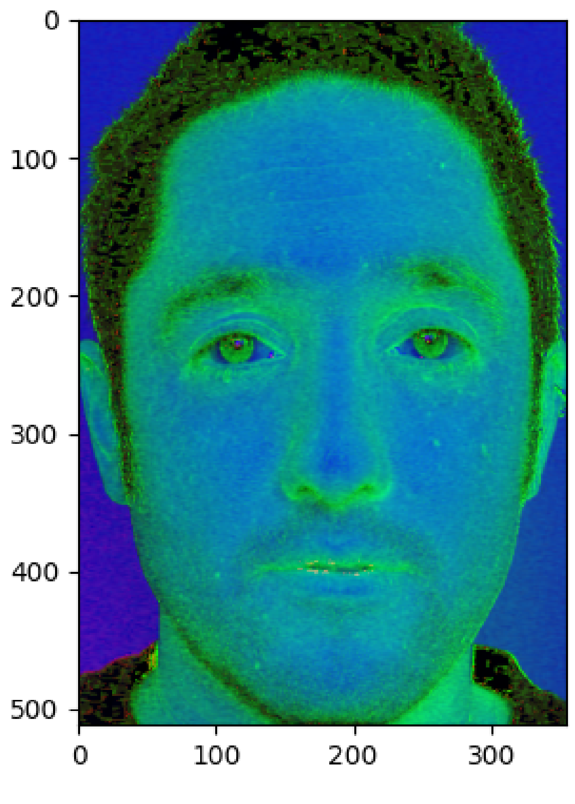
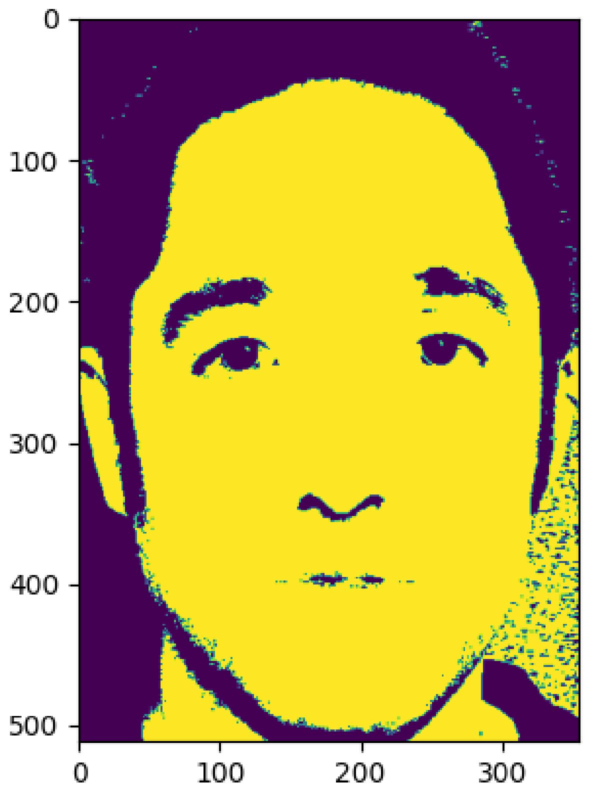
```
lower_nose = np.array([7,47,72],np.uint8)

upper_nose = np.array([13,181,163],np.uint8)

nose = cv2.inRange(img_hsv, lower_nose, upper_nose)
```

	
Before Thresholding	After Thresholding

2.4 Skin

I found the skin part by using the thresholding in the HSV image. I set a suitable range for the skin color in HSV coordinates. Next, I threshold the HSV image using the background color range specified.

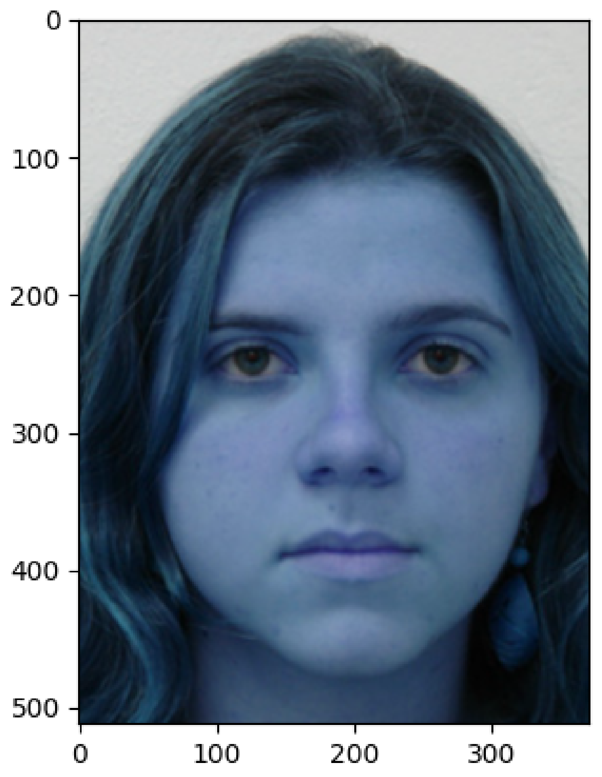
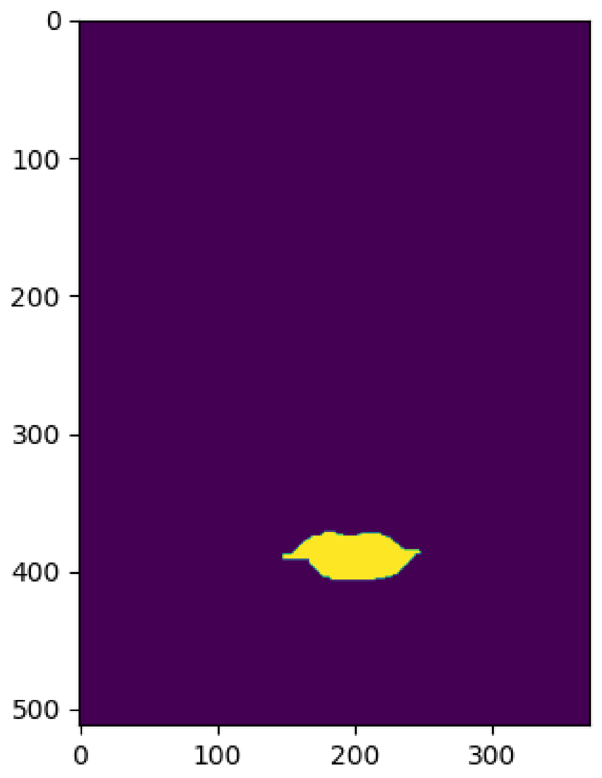
```
lower_skin = np.array([0,48,80],np.uint8)
upper_skin = np.array([20,255,255],np.uint8)
skin = cv2.inRange(img_hsv, lower_skin, upper_skin)
```

	
Before Thresholding	After Thresholding

2.5 Mouth

I found the mouth part by using GrabCut Algorithm. I create a similar mask image and specify which areas are foreground and background. Besides, I also create the coordinates of the mouth which includes the foreground object in the format. The mask image will be modified when run the grabcut. In the new mask image (‘mouth’), pixels will be marked with four flags denoting background and foreground as specified above. So we modify the mask such that all 0-pixels and 2-pixels are put to background and all 1-pixels and 3-pixels are put to foreground pixels.

```
mask = np.zeros(img.shape[:2],np.uint8)
bgdModel = np.zeros((1,65),np.float64)
fgdModel = np.zeros((1,65),np.float64)
rect = (100,300,150,150)
cv2.grabCut(img,mask,rect,bgdModel,fgdModel,5,cv2.GC_INIT_WITH_RECT)
mouth = np.where((mask==2)|(mask==0),0,1).astype('uint8')
```

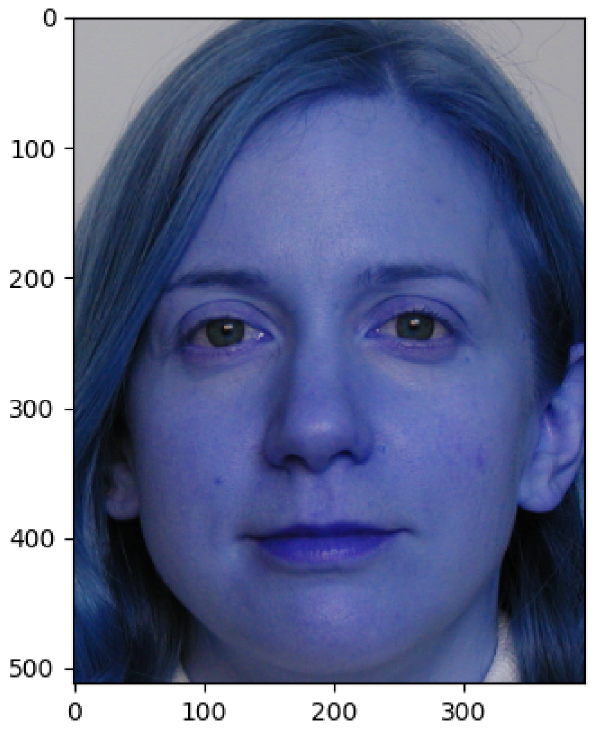
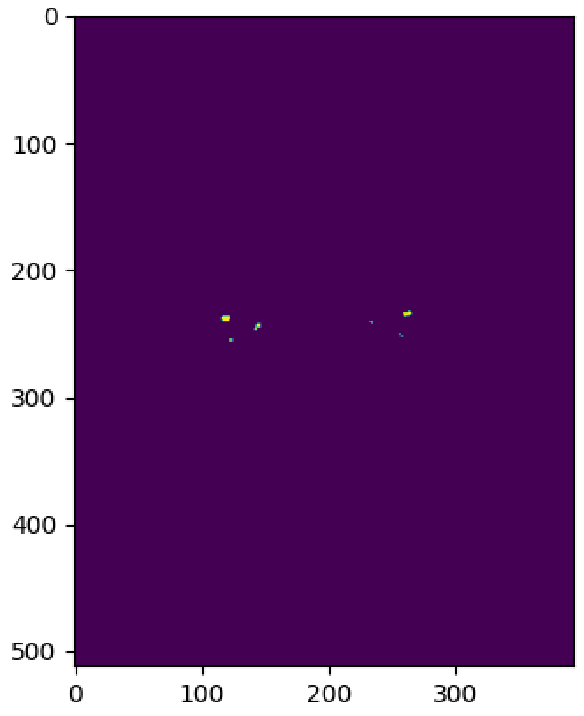
	
Before Processing	After Processing

2.6 Eyes

I found the eyes part by using the GrabCut Algorithm. I create a similar mask image and specify which areas are foreground and background. Besides, I also create the coordinates of the mouth which includes the foreground object in the format. The mask image will be modified when run the grabcut. In the new mask image (‘eye’), pixels will be marked with four flags denoting background and foreground as specified above. So we modify the mask such that all 0-pixels and 2-pixels are put to background and all 1-pixels and 3-pixels are put to foreground pixels.

```
mask = np.zeros(img.shape[:2],np.uint8)
```

```
bgdModel = np.zeros((1,65),np.float64)

fgdModel = np.zeros((1,65),np.float64)

rect = (50,200,250,100)

cv2.grabCut(img,mask,rect,bgdModel,fgdModel,5,cv2.G

C_INIT_WITH_RECT)

eye =

np.where((mask==2)|(mask==0),0,1).astype('uint8')
```

 0 100 200 300 400 500 0 100 200 300	
Before Processing	After Processing

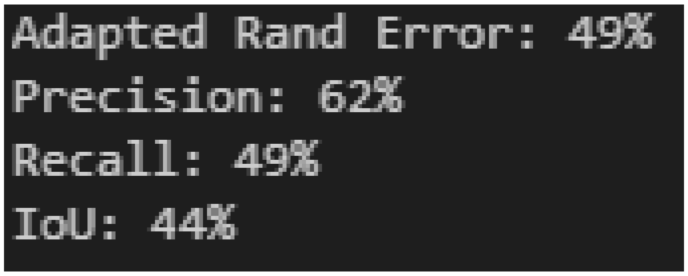
2.7 Combine and Set Intensity

Lastly, I combine the background part, hair and eyebrows part and mouth part, eyes part, nose part and skin part as one image. After that, I set the intensity of the images with the specified number.

```
outImg = bcg + hair + mouth + eye + nose + skin


outImg[np.where(bcg)] = [0]

outImg[np.where(hair)] = [1]

outImg[np.where(mouth)] = [2]

outImg[np.where(eye)] = [3]

outImg[np.where(nose)] = [4]

outImg[np.where(skin)] = [5]
```

3.0 Results & Analysis

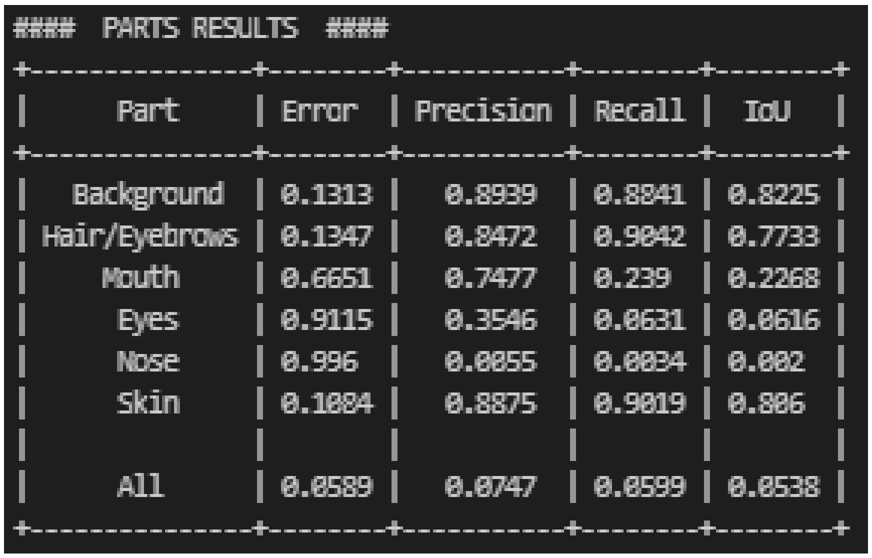
My algorithm for this assignment is getting 49% Adapted Rand Error, 62% Precision. 49% Recall and 44% IoU.



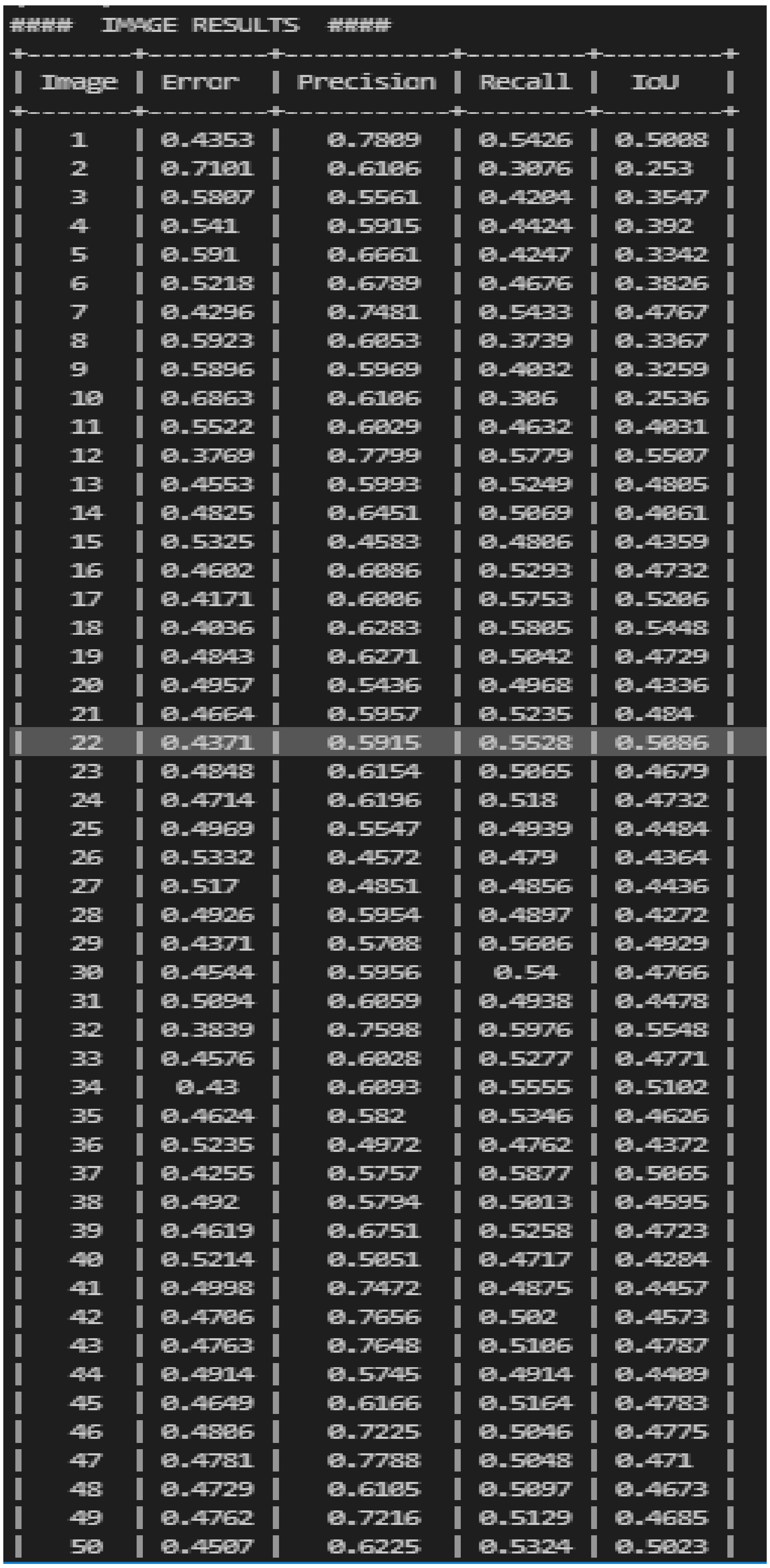
```
Adapted Rand Error: 49%
Precision: 62%
Recall: 49%
IoU: 44%
```

Based on the pictures below, the background part, hair and eyebrows part and skin part obtain a low score in error and high score in precision, recall and IoU. However, the eyes part and nose part obtain a high score in error and low score in precision, recall and IoU. The mouth part obtained a high score in error and precision,but low score in recall and IoU.

The background part, hair and eyebrows part and skin part will get the better performance because the background, hair, eyebrows and skin is having a similar color between each image, so they can exactly segment by using thresholding in HSV colorspace. Nevertheless, the nose part is no good because the color of the nose is very similar with skin and makes it hardly segment by using thresholding in HSV colorspace. Besides, the reason for the bad performance in mouth part and eyes part is the coordinates of mouth and eye may different between each other.



```
####  PARTS RESULTS  ####
+---------------+--------+-----------+--------+--------+
|      Part     | Error  | Precision | Recall |  IoU   |
+---------------+--------+-----------+--------+--------+
|   Background  | 0.1313 |   0.8939  | 0.8841 | 0.8225 |
| Hair/Eyebrows | 0.1347 |   0.8472  | 0.9042 | 0.7733 |
|     Mouth     | 0.6651 |   0.7477  | 0.239  | 0.2268 |
|      Eyes     | 0.9115 |   0.3546  | 0.0631 | 0.0616 |
|      Nose     | 0.996  |   0.0855  | 0.0034 | 0.002  |
|      Skin     | 0.1084 |   0.8875  | 0.9019 | 0.806  |
|               |        |           |        |        |
|      All      | 0.0589 |   0.0747  | 0.0599 | 0.0538 |
+---------------+--------+-----------+--------+--------+
```



```
####  IMAGE RESULTS  ####
+--------+--------+------------+--------+--------+
| Image  | Error  | Precision  | Recall |  IoU   |
+--------+--------+------------+--------+--------+
|   1    | 0.4353 |   0.7809   | 0.5426 | 0.5008 |
|   2    | 0.7101 |   0.6106   | 0.3076 | 0.253  |
|   3    | 0.5807 |   0.5561   | 0.4204 | 0.3547 |
|   4    | 0.541  |   0.5915   | 0.4424 | 0.392  |
|   5    | 0.591  |   0.6661   | 0.4247 | 0.3342 |
|   6    | 0.5218 |   0.6789   | 0.4676 | 0.3826 |
|   7    | 0.4296 |   0.7481   | 0.5433 | 0.4767 |
|   8    | 0.5923 |   0.6853   | 0.3739 | 0.3367 |
|   9    | 0.5896 |   0.5969   | 0.4032 | 0.3259 |
|   10   | 0.6863 |   0.6106   | 0.306  | 0.2536 |
|   11   | 0.5522 |   0.6029   | 0.4632 | 0.4031 |
|   12   | 0.3769 |   0.7799   | 0.5779 | 0.5507 |
|   13   | 0.4553 |   0.5993   | 0.5249 | 0.4805 |
|   14   | 0.4825 |   0.6451   | 0.5069 | 0.4061 |
|   15   | 0.5325 |   0.4583   | 0.4806 | 0.4359 |
|   16   | 0.4602 |   0.6086   | 0.5293 | 0.4732 |
|   17   | 0.4171 |   0.6006   | 0.5753 | 0.5206 |
|   18   | 0.4036 |   0.6283   | 0.5805 | 0.5448 |
|   19   | 0.4843 |   0.6271   | 0.5042 | 0.4729 |
|   20   | 0.4957 |   0.5436   | 0.4968 | 0.4336 |
|   21   | 0.4664 |   0.5957   | 0.5235 | 0.484  |
|   22   | 0.4371 |   0.5915   | 0.5528 | 0.5086 |
|   23   | 0.4848 |   0.6154   | 0.5065 | 0.4679 |
|   24   | 0.4714 |   0.6196   | 0.518  | 0.4732 |
|   25   | 0.4969 |   0.5547   | 0.4939 | 0.4484 |
|   26   | 0.5332 |   0.4572   | 0.479  | 0.4364 |
|   27   | 0.517  |   0.4851   | 0.4856 | 0.4436 |
|   28   | 0.4926 |   0.5954   | 0.4897 | 0.4272 |
|   29   | 0.4371 |   0.5708   | 0.5606 | 0.4929 |
|   30   | 0.4544 |   0.5956   |  0.54  | 0.4766 |
|   31   | 0.5094 |   0.6859   | 0.4938 | 0.4478 |
|   32   | 0.3839 |   0.7598   | 0.5976 | 0.5548 |
|   33   | 0.4576 |   0.6028   | 0.5277 | 0.4771 |
|   34   |  0.43  |   0.6093   | 0.5555 | 0.5102 |
|   35   | 0.4624 |   0.582    | 0.5346 | 0.4626 |
|   36   | 0.5235 |   0.4972   | 0.4762 | 0.4372 |
|   37   | 0.4255 |   0.5757   | 0.5877 | 0.5065 |
|   38   | 0.492  |   0.5794   | 0.5013 | 0.4595 |
|   39   | 0.4619 |   0.6751   | 0.5258 | 0.4723 |
|   40   | 0.5214 |   0.5851   | 0.4717 | 0.4284 |
|   41   | 0.4998 |   0.7472   | 0.4875 | 0.4457 |
|   42   | 0.4706 |   0.7656   | 0.502  | 0.4573 |
|   43   | 0.4763 |   0.7648   | 0.5106 | 0.4787 |
|   44   | 0.4914 |   0.5745   | 0.4914 | 0.4409 |
|   45   | 0.4649 |   0.6166   | 0.5164 | 0.4783 |
|   46   | 0.4806 |   0.7225   | 0.5046 | 0.4775 |
|   47   | 0.4781 |   0.7788   | 0.5048 | 0.471  |
|   48   | 0.4729 |   0.6105   | 0.5097 | 0.4673 |
|   49   | 0.4762 |   0.7216   | 0.5129 | 0.4685 |
|   50   | 0.4507 |   0.6225   | 0.5324 | 0.5023 |
```

4.0 Suggestions for Improvement

The suggestion for improvement is using DeepLab. This is because DeepLab can perform image segmentation while helping to control signal extraction. Beside, it also enables multi-scale contextual feature learning and aggregate features from images of different scales. DeepLab uses ImageNet pre-trained ResNet for feature extraction. DeepLab uses hole convolution instead of regular convolution. The different expansion rate of each convolution enables the ResNet block to capture multi-scale context information.