Abstract

With the rise of capturing pictures and video technology, more digital image and video data are available for people to process, explore and utilize. In this assignment, an algorithm composed of six functions is designed to automatically segment the face image into multiple parts. The basic image processing techniques used are color space conversion, color filtering, edge detection and morphological operations. An image is divided into six parts: background, hair and eyebrows, mouth, eyes, nose, and skin. The algorithm achieved 65% pixel wise precision, 62% pixel wise recall, 46% Intersection over Union (IoU) and had 40% adapted rand error on a set of 50 face images.

Keywords: face segmentation, facial features segmentation, color space conversion, color filtering, edge detection, morphological operations, erosion, dilation

1.0 Introduction

Computer vision is a popular field under the artificial intelligence domain, which trains computers and systems to view and understand digital images and videos. This field is often considered as high-level image processing because it usually involves extracting information and generating insights from the digital images and videos.

However, this assignment focuses more on mid-level image processing, which involves image segmentation by implementing basic image processing methods. The specific task given is to use a dataset containing 50 human frontal face images and ground truth images to segment facial features from the images. The required segmentation parts are the background, hair and eyebrows, mouth, eyes, nose, and skin. The expected outcome is grayscale images with different intensities for different segmented parts. An evaluation of adaptive rand error, pixel wise precision and recall, and the IoU between the generated output segmentation and the corresponding ground truth segmentation will be computed. In order to obtain a more general view of the result, the average evaluations for the output and average evaluations of each face part will also be calculated. These values helps to explain the generated outputs in a more detailed way.

The motivation for this assignment is to learn the difference in visual perception between computers and humans by approaching from a low-level perspective. Computers view the image as each pixel, while humans view the image as a whole and interpret the information contained in the image, such as object, color, quality, etc. This assignment allows us to think about how to use low-level features, such as edges, colors, and textures on the image, so that the computers can interpret the images like a human. In addition, we may find the incomprehensible parts of digital images from a computer perspective and understand the importance of deep learning in image processing.

The possible applications of face segmentation include, but are not limited to, face retouching and beautification on images, virtual cosmetics wearing on faces, virtual hair dyeing effects, birthmarks and moles spotting. Apart from this, face segmentation can also help complete demographic tasks such as age, gender, and ethnicity classification. With enough digital images or videos and advanced face segmentation technology, we are able to create new fake faces by obtaining different facial features from different inputs, and calculate the most common face shapes, hair colors, eyes sizes, nose heights, lips thickness, etc.

2.0 Description of Methods

In this assignment, the algorithm is designed based on the observations on the dataset of 50 human frontal face images and coded in Python language thus stored with the filename, *imageSegment.py*. It consisted of six different functions for six different segmentation parts, color space conversions for processing the input image and output image, and pixel intensities allocations.

The image processing techniques such as canny edge detection, color filtering, color space conversion, and morphological operations are implemented in the algorithm and used repeatedly in different segmentation functions. The Canny edge detector uses a multi-stage algorithm to detect edges in image. The color filtering technique is used along with different ranges of color to mask out the desired regions based on colors. Color space conversion is considered as the image pre-processing step to obtain image in the most suitable color space for further processing. Morphological operations including erosion, dilation, opening and closing. Opening is the combined processes of erosion followed by dilation whereas closing is the combined processes of dilation followed by erosion. Erosion will remove pixels on objects boundaries which are the small anomalies while dilation will add pixels to the boundaries of objects in image which holes and broken areas are filled. Erosion will decrease the size of the objects while dilation will increase the size of the objects. The concept of region of interest (ROI) is also used in the functions to prevent the output mask influenced by unwanted areas of similar colors.

First of all, the algorithm will take input of one image at once with a 3-dimensional numpy array in BGR format. The image will be converted into different color spaces with Python OpenCV, *cv2.cvtColor()* method thus be used in different segmentation functions. The color spaces applied are grayscale, Hue, Saturation, Value (HSV), CIELAB (L*a*b), YCbCr, and CIE 1931 XYZ (XYZ). The converted images will be saved with different variable names corresponding to their color spaces.

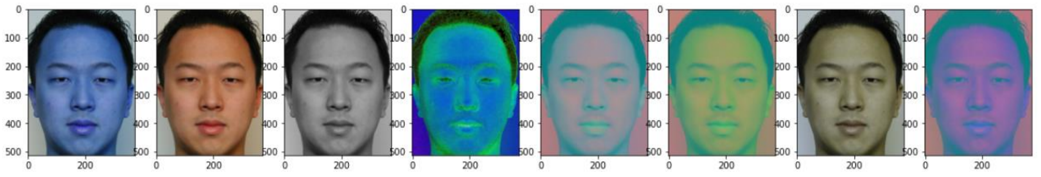


Figure 1: Same Image in Different Color Spaces (04.bmp)

Next, the converted image will be parsed into different segmentation functions. *background_segmentation()*, *hairEyebrows_segmentation()*, *mouth_segmentation()*, *eyes_segmentation()*, *nose_segmentation()*, and *skin_segmentation()*. Different functions used different color spaces-converted images based on the observations and suitability of the specific segmentation part with the images.

The **background** segmentation used image in **YCbCr** color space. This is because YCbCr image give a view of the background colors are quite distinct from other parts. A mask is obtained from applying color filtering within a range of colors that is similar to the background colors. A cross-shaped kernel is created with *cv2.getStructuringElement()* and *cv2.MORPH_CROSS* methods. Thus, *cv2.bitwise_and()* method is applied on the image with the mask created earlier. The output is then experienced dilation followed by erosion with the usage of the kernel, *cv2.morphologyEx()* and *cv2.MORPH_CLOSE* methods. This function will ultimately return the output mask in RGB format.

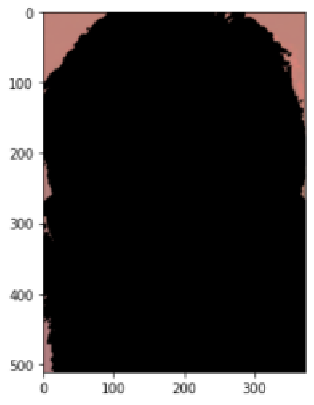


Figure 2: Background Segmentation (04.bmp)

The **hair** and **eyebrows** segmentation used image in **XYZ** color space, then an inverted triangle shaped white patch is overlaid on the middle-bottom of the image. This is because XYZ image give a view of the hair and eyebrows colors are quite distinct from other parts except the eyes and darker shadows on face, the white patch is to block the possible wrongly detected area. A mask is obtained from applying color filtering within a range of colors that is similar to the hair and eyebrows colors. Thus, *cv2.bitwise_and()* method is applied on the image with the mask created earlier. This function will ultimately return the output mask in RGB format.

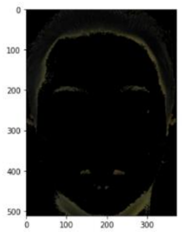


Figure 3: Hair and Eyebrows Segmentation (04.bmp)

The **mouth** segmentation used image in **L*a*b** color space, then the ROI for the mouth area is cropped and a copy of the input image converted all its pixels into white. The ROI of mouth is overlaid on the copied image at the same position of ROI to produce a new image that is white color except the mouth area. This is because L*a*b image give a view of the mouth colors are quite distinct from skin, by cropping the ROI out, the filtering can focus on the estimated correct area. Thus, a mask is obtained from applying color filtering within a range of colors that is similar to the mouth colors. An elliptical kernel is created with *cv2.getStructuringElement()* and *cv2.MORPH_ELLIPSE* methods, then *cv2.bitwise_and()* method is applied on the image with the mask created earlier. The output is then experienced erosion followed by dilation with the usage of the kernel, *cv2.erode()* and *cv2.dilate()* methods. This function will ultimately return the output mask in RGB format.

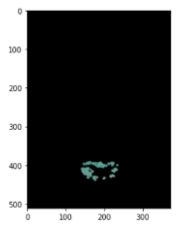


Figure 4: Mouth Segmentation (04.bmp)

The **eyes** segmentation used image in **grayscale** color space, then Canny edge detection (*cv2.Canny()*) is used to obtain a new image output with only the edges. A ROI for the eyes area is cropped and a copy of the input image converted all its pixels into white. The ROI of eyes is overlaid on the copied image at the same position of ROI to produce a new image that is white color except the eyes area. This is because using edges can get better eyes shapes, by cropping the ROI out, the other strong edges can be ignored. Thus, a mask is obtained from applying color filtering within a range of colors that is similar to the eyes colors. An elliptical kernel is created with *cv2.getStructuringElement()* and *cv2.MORPH_ELLIPSE* methods, then *cv2.bitwise_and()* method is applied on the image with the mask created earlier. The output is then experienced dilation followed by erosion with the usage of the kernel, *cv2.morphologyEx()* and *cv2.MORPH_CLOSE* methods. This function will convert the output mask into RGB format and return it.

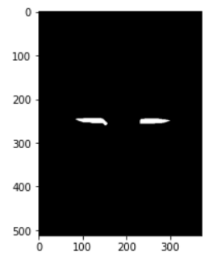


Figure 5: Mouth Segmentation (04.bmp)

The **nose** segmentation used image in **HSV** color space, then a ROI for the nose area is cropped and a copy of the input image converted all its pixels into white. The ROI of nose is overlaid on the copied image at the same position of ROI to produce a new image that is white color except the nose area. This is because HSV image give a view of the nose colors are quite distinct from skin, by cropping the ROI out, the filtering can focus on the estimated correct area. Thus, a mask is obtained from applying color filtering within a range of colors that is similar to the nose colors. A cross-shaped kernel is created with *cv2.getStructuringElement()* and *cv2.MORPH_CROSS* methods, then *cv2.bitwise_and()* method is applied on the image with the mask created earlier. The output is then experienced erosion followed by dilation with the usage of the kernel, *cv2.erode()* and *cv2.dilate()* methods. This function will ultimately return the output mask in RGB format.

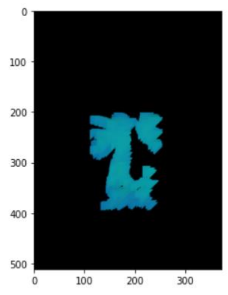


Figure 6: Nose Segmentation (04.bmp)

The **skin** segmentation used image in **HSV** color space, then a mask is obtained from applying color filtering within a range of colors that is similar to the skin colors. An elliptical kernel is created with *cv2.getStructuringElement()* and *cv2.MORPH_ELLIPSE* methods. Thus, *cv2.bitwise_and()* method is applied on the image with the mask created earlier. The output is then experienced erosion followed by dilation with the usage of the kernel, *cv2.erode()* and *cv2.dilate()* methods. This function will ultimately return the output mask in RGB format.

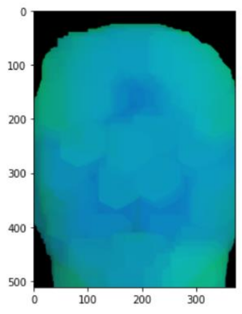


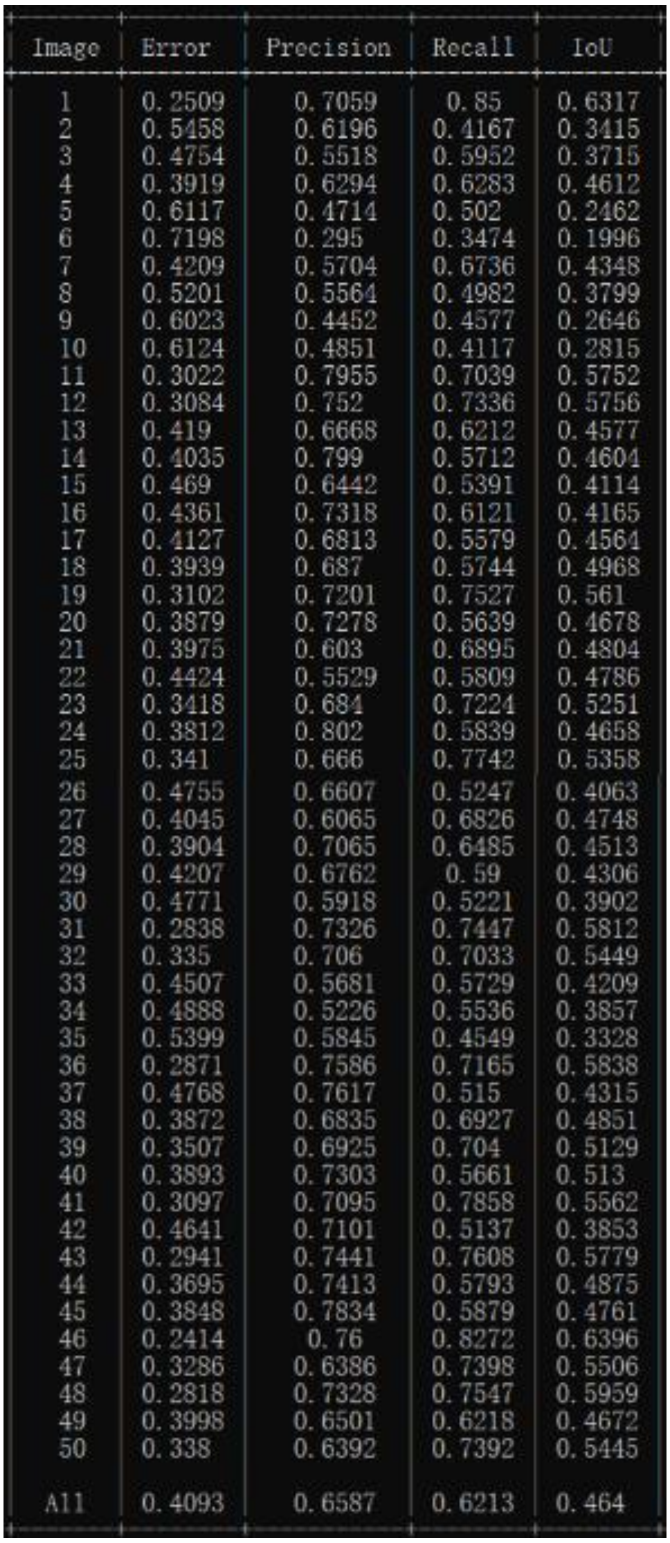
Figure 7: Skin Segmentation (04.bmp)

Finally, all the face segmentation masks are combined together then converted into grayscale color space. The different segmented parts in the newly combined grayscale image are then allocated with respective pixels intensities by using *numpy.where()* function. The skin intensity is 5, nose intensity is 4, eyes intensity is 3, mouth intensity is 2, hair and eyebrows intensity is 1, and background intensity is 0. The lower the intensity, the darker the color in grayscale and vice versa. The final output of the whole algorithm or function is a grayscale image with different intensities in different parts.

3.0 Results & Analysis

The overall avearge results of the algorithm obtained are 40% adapted rand error, 65% pixel wise precision, 62% pixel wise recall, and 46% Intersection over Union (IoU) based on the **human frontal face dataset**. The following are the detailed results and example output images.

Figure 8: Human Frontal Face Dataset Result for Each Image



```
+-------+--------+-----------+--------+--------+
| Image | Error  | Precision | Recall |  IoU   |
+-------+--------+-----------+--------+--------+
|   1   | 0.2509 |   0.7059  |  0.85  | 0.6317 |
|   2   | 0.5458 |   0.6196  | 0.4167 | 0.3415 |
|   3   | 0.4754 |   0.5518  | 0.5952 | 0.3715 |
|   4   | 0.3919 |   0.6294  | 0.6283 | 0.4612 |
|   5   | 0.6117 |   0.4714  | 0.502  | 0.2462 |
|   6   | 0.7198 |   0.295   | 0.3474 | 0.1996 |
|   7   | 0.4209 |   0.5704  | 0.6736 | 0.4348 |
|   8   | 0.5201 |   0.5564  | 0.4982 | 0.3799 |
|   9   | 0.6023 |   0.4452  | 0.4577 | 0.2646 |
|  10   | 0.6124 |   0.4851  | 0.4117 | 0.2815 |
|  11   | 0.3022 |   0.7955  | 0.7039 | 0.5752 |
|  12   | 0.3084 |   0.752   | 0.7336 | 0.5756 |
|  13   | 0.419  |   0.6668  | 0.6212 | 0.4577 |
|  14   | 0.4035 |   0.799   | 0.5712 | 0.4604 |
|  15   | 0.469  |   0.6442  | 0.5391 | 0.4114 |
|  16   | 0.4361 |   0.7318  | 0.6121 | 0.4165 |
|  17   | 0.4127 |   0.6813  | 0.5579 | 0.4564 |
|  18   | 0.3939 |   0.687   | 0.5744 | 0.4968 |
|  19   | 0.3102 |   0.7201  | 0.7527 | 0.561  |
|  20   | 0.3879 |   0.7278  | 0.5639 | 0.4678 |
|  21   | 0.3975 |   0.603   | 0.6895 | 0.4804 |
|  22   | 0.4424 |   0.5529  | 0.5809 | 0.4786 |
|  23   | 0.3418 |   0.684   | 0.7224 | 0.5251 |
|  24   | 0.3812 |   0.802   | 0.5839 | 0.4658 |
|  25   | 0.341  |   0.666   | 0.7742 | 0.5358 |
|  26   | 0.4755 |   0.6607  | 0.5247 | 0.4063 |
|  27   | 0.4045 |   0.6065  | 0.6826 | 0.4748 |
|  28   | 0.3904 |   0.7065  | 0.6485 | 0.4513 |
|  29   | 0.4207 |   0.6762  |  0.59  | 0.4306 |
|  30   | 0.4771 |   0.5918  | 0.5221 | 0.3902 |
|  31   | 0.2838 |   0.7326  | 0.7447 | 0.5812 |
|  32   | 0.335  |   0.706   | 0.7033 | 0.5449 |
|  33   | 0.4507 |   0.5681  | 0.5729 | 0.4209 |
|  34   | 0.4888 |   0.5226  | 0.5536 | 0.3857 |
|  35   | 0.5399 |   0.5845  | 0.4549 | 0.3328 |
|  36   | 0.2871 |   0.7586  | 0.7165 | 0.5838 |
|  37   | 0.4768 |   0.7617  | 0.515  | 0.4315 |
|  38   | 0.3872 |   0.6835  | 0.6927 | 0.4851 |
|  39   | 0.3507 |   0.6925  | 0.704  | 0.5129 |
|  40   | 0.3893 |   0.7303  | 0.5661 | 0.513  |
|  41   | 0.3097 |   0.7095  | 0.7858 | 0.5562 |
|  42   | 0.4641 |   0.7101  | 0.5137 | 0.3853 |
|  43   | 0.2941 |   0.7441  | 0.7608 | 0.5779 |
|  44   | 0.3695 |   0.7413  | 0.5793 | 0.4875 |
|  45   | 0.3848 |   0.7834  | 0.5879 | 0.4761 |
|  46   | 0.2414 |    0.76   | 0.8272 | 0.6396 |
|  47   | 0.3286 |   0.6386  | 0.7398 | 0.5506 |
|  48   | 0.2818 |   0.7328  | 0.7547 | 0.5959 |
|  49   | 0.3998 |   0.6501  | 0.6218 | 0.4672 |
|  50   | 0.338  |   0.6392  | 0.7392 | 0.5445 |
|       |        |           |        |        |
|  All  | 0.4093 |   0.6587  | 0.6213 | 0.464  |
+-------+--------+-----------+--------+--------+
```

The adapted Rand error is the frequency at which the ground truth areas disagree over segmented areas on whether a pair of pixels belongs to same or different parts. The algorithm has a 40% adapted Rand error, which means that for every 100 pixels, there were 40 pixels are wrongly segmented and do not conform to the ground truth.

The pixel wise precision is the ratio of the number of true positives to the total number of positive segmentations by image pixels. The algorithm achieved a 65% precision, which means that for every 100 pixels, there were 65 pixels correctly segmented out of all the segmented pixels.

The pixel wise recall is the ratio of the number of true positives to the total number of both correct positive and negative segmentations by images pixels. The algorithm reached a 62% recall, which means that for every 100 pixels, there were 62 pixels correctly segmented out of all the ground truth pixels.

The IoU is the ratio between the intersection and the union of the segmented areas and ground truth areas. The algorithm achieved a 46% IoU which means only 46% of areas of segmentations are correctly overlapped with all the ground truth areas.

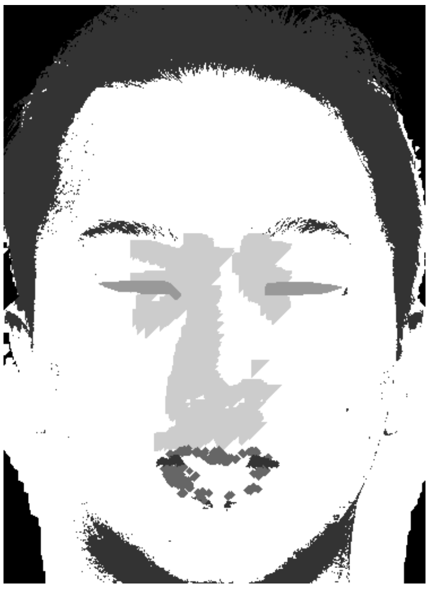


Figure 9: Human Frontal Face Output 04

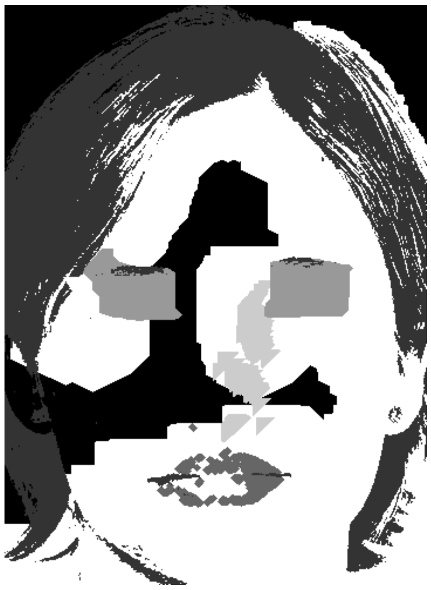


Figure 10: Human Frontal Face Output 07

According to the figures shown above, we can see that applying the same algorithm to different images can produce very different outputs. This is because only using one algorithm is impossible to suit every images tested as the images have different properties and characteristics.

Table 1: Human Frontal Face Dataset Result for Each Segmentation Part

Part	**Error**	**Precision**	**Recall**	**IoU**
Background	0.2376	0.7955	0.7663	0.6694
Hair/Eyebrows	0.1977	0.7459	0.8942	0.6820
Mouth	0.5834	0.6509	0.3426	0.2755
Eyes	0.5511	0.6317	0.4413	0.3137
Nose	0.6371	0.2966	0.5865	0.233
Skin	0.2493	0.8315	0.6967	0.6106
All	**0.0491**	**0.0790**	**0.0746**	**0.0557**

By looking at the detailed results for each segmentation part, we can have a better understanding on each segmentation part.

For background segmentation, the precision is 79% and recall is 76%. The overall background colors for all images are quite similar but still some are different. The possible factor influenced the result is the different lightning used when taking the images because this will affect the overall tones, light and shades of the images. Thus this will resulting inconsistency in color filtering and edge detections.

For hair and eyebrows segmentation, the precision is 74% and recall is 89%. Most of the images given have dark hair and eyebrows. The possible factors influenced the result are the different hair and eyebrows colors and the eyes being wrongly segmented.

For mouth segmentation, the precision is 65% and recall is 34%. The possible factor influenced the result are the different mouth shapes and sizes, the ROI of mouth is not accurate for every image.

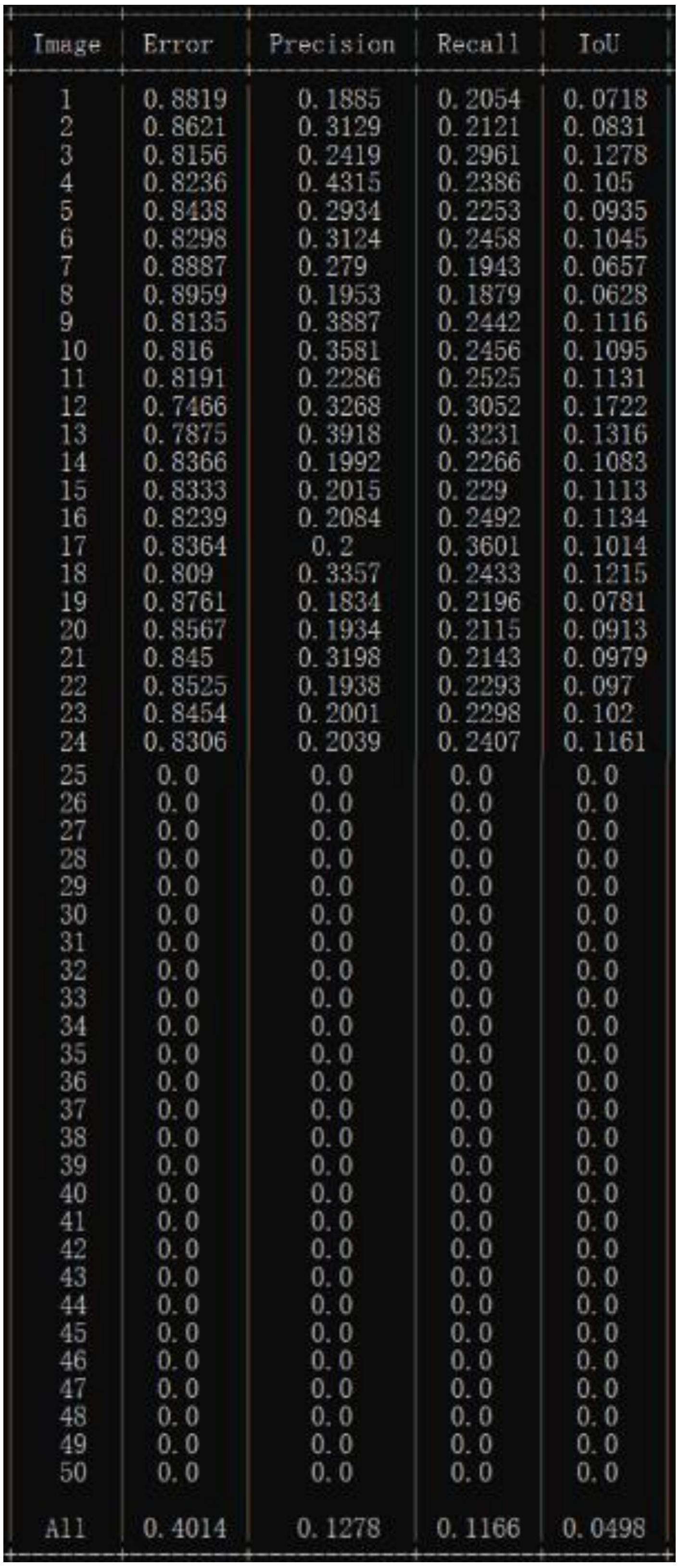
For eyes segmentation, the precision is 63% and recall is 44%. The possible factor influenced the result are the different eyes shapes and sizes, the ROI of eyes is not accurate for every image.

For nose segmentation, the precision is 29% and recall is 58%. The possible factor influenced the result are the different nose shapes and sizes, the ROI of nose is not accurate for every image, the nose colors are too similar to the skin.

For skin segmentation, the precision is 83% and recall is 67%. The possible factor influenced the result are the different face shapes, size, complexions and colors for different ethnicity of people. The blemishes, scars, moles, makeup might affect the result because these spots have different colors from the skin.

The overall average results of the algorithm obtained are 40% adapted rand error, 12% pixel wise precision, 11% pixel wise recall, and 4% Intersection over Union (IoU) based on the **human multi-angle face dataset**. The following are the detailed results and example output images.

Figure 11: Human Multi-angle Face Dataset Result for Each Image



```
+-------+--------+-----------+--------+--------+
| Image | Error  | Precision | Recall |  IoU   |
+-------+--------+-----------+--------+--------+
|   1   | 0.8819 |   0.1885  | 0.2054 | 0.0718 |
|   2   | 0.8621 |   0.3129  | 0.2121 | 0.0831 |
|   3   | 0.8156 |   0.2419  | 0.2961 | 0.1278 |
|   4   | 0.8236 |   0.4315  | 0.2386 | 0.105  |
|   5   | 0.8438 |   0.2934  | 0.2253 | 0.0935 |
|   6   | 0.8298 |   0.3124  | 0.2458 | 0.1045 |
|   7   | 0.8887 |   0.279   | 0.1943 | 0.0657 |
|   8   | 0.8959 |   0.1953  | 0.1879 | 0.0628 |
|   9   | 0.8135 |   0.3887  | 0.2442 | 0.1116 |
|   10  | 0.816  |   0.3581  | 0.2456 | 0.1095 |
|   11  | 0.8191 |   0.2286  | 0.2525 | 0.1131 |
|   12  | 0.7466 |   0.3268  | 0.3052 | 0.1722 |
|   13  | 0.7875 |   0.3918  | 0.3231 | 0.1316 |
|   14  | 0.8366 |   0.1992  | 0.2266 | 0.1083 |
|   15  | 0.8333 |   0.2015  | 0.229  | 0.1113 |
|   16  | 0.8239 |   0.2084  | 0.2492 | 0.1134 |
|   17  | 0.8364 |    0.2    | 0.3601 | 0.1014 |
|   18  | 0.809  |   0.3357  | 0.2433 | 0.1215 |
|   19  | 0.8761 |   0.1834  | 0.2196 | 0.0781 |
|   20  | 0.8567 |   0.1934  | 0.2115 | 0.0913 |
|   21  | 0.845  |   0.3198  | 0.2143 | 0.0979 |
|   22  | 0.8525 |   0.1938  | 0.2293 | 0.097  |
|   23  | 0.8454 |   0.2001  | 0.2298 | 0.102  |
|   24  | 0.8306 |   0.2039  | 0.2407 | 0.1161 |
|   25  |  0.0   |    0.0    |  0.0   |  0.0   |
|   26  |  0.0   |    0.0    |  0.0   |  0.0   |
|   27  |  0.0   |    0.0    |  0.0   |  0.0   |
|   28  |  0.0   |    0.0    |  0.0   |  0.0   |
|   29  |  0.0   |    0.0    |  0.0   |  0.0   |
|   30  |  0.0   |    0.0    |  0.0   |  0.0   |
|   31  |  0.0   |    0.0    |  0.0   |  0.0   |
|   32  |  0.0   |    0.0    |  0.0   |  0.0   |
|   33  |  0.0   |    0.0    |  0.0   |  0.0   |
|   34  |  0.0   |    0.0    |  0.0   |  0.0   |
|   35  |  0.0   |    0.0    |  0.0   |  0.0   |
|   36  |  0.0   |    0.0    |  0.0   |  0.0   |
|   37  |  0.0   |    0.0    |  0.0   |  0.0   |
|   38  |  0.0   |    0.0    |  0.0   |  0.0   |
|   39  |  0.0   |    0.0    |  0.0   |  0.0   |
|   40  |  0.0   |    0.0    |  0.0   |  0.0   |
|   41  |  0.0   |    0.0    |  0.0   |  0.0   |
|   42  |  0.0   |    0.0    |  0.0   |  0.0   |
|   43  |  0.0   |    0.0    |  0.0   |  0.0   |
|   44  |  0.0   |    0.0    |  0.0   |  0.0   |
|   45  |  0.0   |    0.0    |  0.0   |  0.0   |
|   46  |  0.0   |    0.0    |  0.0   |  0.0   |
|   47  |  0.0   |    0.0    |  0.0   |  0.0   |
|   48  |  0.0   |    0.0    |  0.0   |  0.0   |
|   49  |  0.0   |    0.0    |  0.0   |  0.0   |
|   50  |  0.0   |    0.0    |  0.0   |  0.0   |
|       |        |           |        |        |
|  All  | 0.4014 |   0.1278  | 0.1166 | 0.0498 |
+-------+--------+-----------+--------+--------+
```

The adapted Rand error is the frequency at which the ground truth areas disagree over segmented areas on whether a pair of pixels belongs to same or different parts. The algorithm has a 40% adapted Rand error, which means that for every 100 pixels, there were 40 pixels are wrongly segmented and do not conform to the ground truth.

The pixel wise precision is the ratio of the number of true positives to the total number of positive segmentations by image pixels. The algorithm achieved a 12% precision, which means that for every 100 pixels, there were 12 pixels correctly segmented out of all the segmented pixels.

The pixel wise recall is the ratio of the number of true positives to the total number of both correct positive and negative segmentations by images pixels. The algorithm reached a 11% recall, which means that for every 100 pixels, there were 11 pixels correctly segmented out of all the ground truth pixels.

The IoU is the ratio between the intersection and the union of the segmented areas and ground truth areas. The algorithm achieved a 4% IoU which means only 4% of areas of segmentations are correctly overlapped with all the ground truth areas.

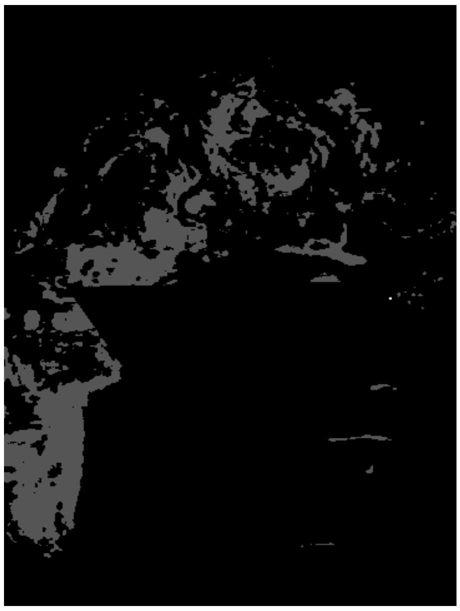


Figure 12: Human Multi-angle Face Output 01

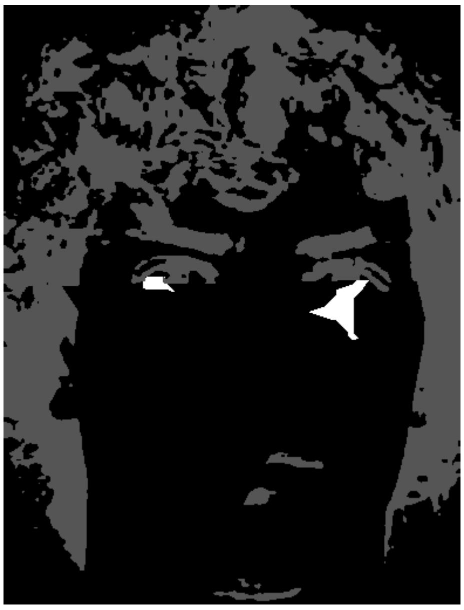


Figure 13: Human Multi-angle Face Output 03

According to the figures shown above, we can see that applying the algorithm designed for frontal face segmentation is not suitable for multi-angle face images. This is mainly because the region of interest pre-set for frontal facing is not suitable for images taken with different angles of faces.

Table 2: Human Multi-angle Face Dataset Result for Each Segmentation Part

Part	Error	Precision	Recall	IoU
Background	0.2979	0.1137	0.4769	0.1135
Hair/Eyebrows	0.2672	0.4474	0.1437	0.1399
Mouth	0.48	0.0	0.0	0.0
Eyes	0.4245	0.0739	0.0671	0.0341
Nose	0.48	0.0	0.0	0.0
Skin	0.4588	0.1317	0.0118	0.0114
All	**0.0482**	**0.0153**	**0.0140**	**0.0060**

As a whole, the current algorithm is not suitable for multi-angle face images, hence the accurate segmentations only happen when the images are frontal facing. There are less frontal facing images than other angles facing images, thus the results obtained for this dataset is very poor. It will be required another algorithm take into considerations of the properties and characteristics of the multi-angle face images.

4.0 Suggestions for Improvement

The algorithm designed in this assignment still has a lot of room for improvements as it cannot produce remarkable and significantly good results. From low-level and mid-level image processing perspectives, the possible improvements are such as pre-processing the images to standardize and balance the color tones through filtering, enhancement, detection of the faces angle, orientation, geometric view, thus direct to different algorithms that suit for the angle or pre-process the images into desired position and angle to suit the one algorithm more. The extraction of textures on hair, eyebrows and calculation on average region of interest of each facial features can be considered too.

Moreover, powerful graph-based algorithms such as Felzenswalb's method and SLIC superpixels can be used to segment regions in a more accurate way.

From high-level image processing perspective which involves the artificial intelligence domain knowledge, we can develop a deep learning model to do the face segmentations tasks. This will give us the most accurate output compared to other methods, as long as we have enough data to train the model. However, this will required more machine power and resources.

With the advancement of technology, there will be other methods to improve face segmentation in the future.