CMPE 537 - Computer Vision

Assignment - 1

Abdullah Atakan Gney

October 14, 2018

Contents

Chapter 1

Task-1

1.1 Determining Color Channels

In this part of the Assignment I have split the original image into RGB channels, and converted color-space to HSV and split into H, S, and V channels. After that, I have plotted both original RGB, and HSV images and I also plotted each channel separately with their intensity values mapped in to range from 0 to 255.

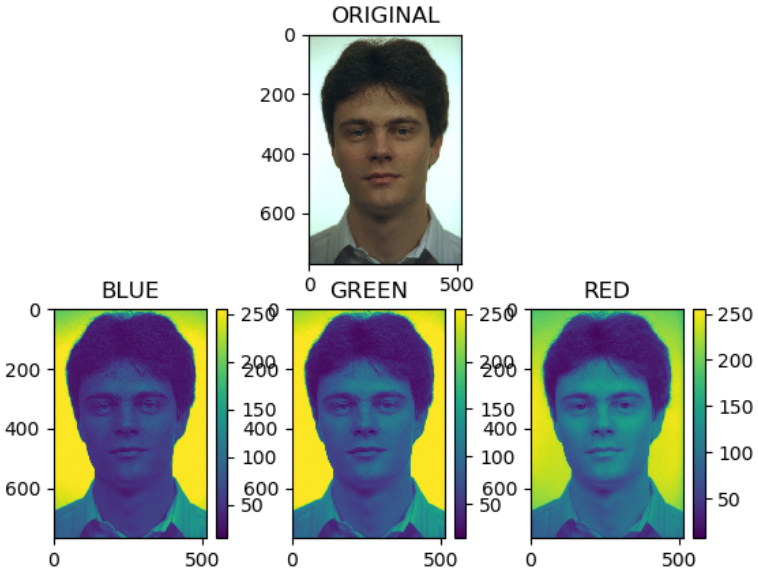


Figure 1.1: RGB plots for img_001.jpg

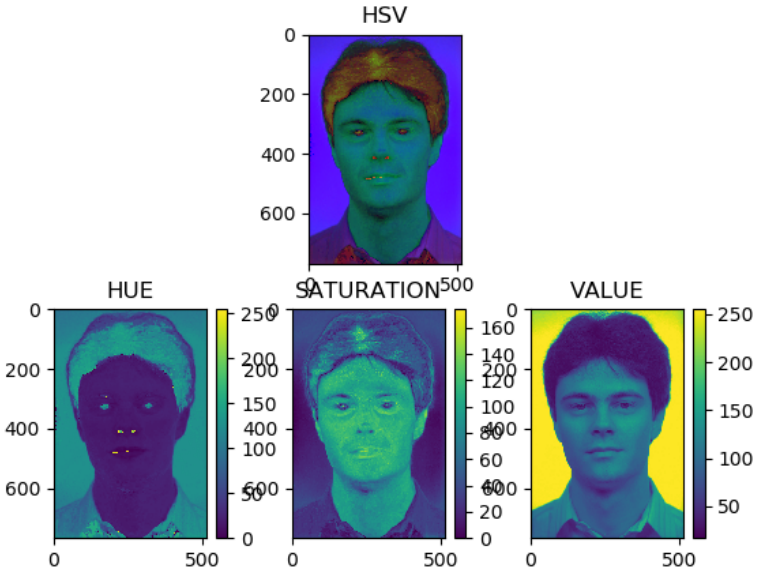
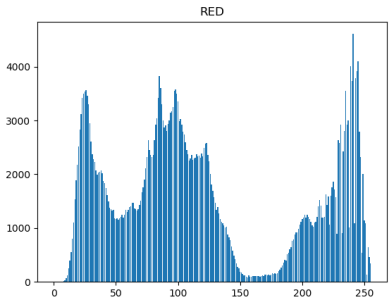


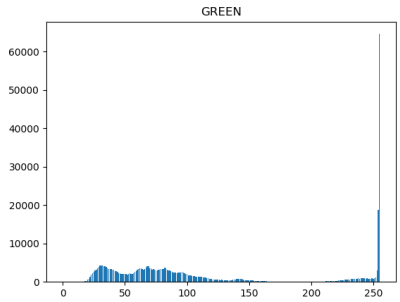
Figure 1.2: HSV plots for img_001.jpg

1.2 Histograms of each channel

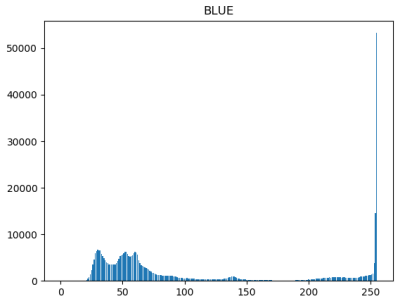
For this part of the task, I have implemented a histogram function, which is based on numpy library. I have also checked my implementation of histogram function with 3 other ways to calculate histograms, one of them is numpy's histogram function.



(a) RED

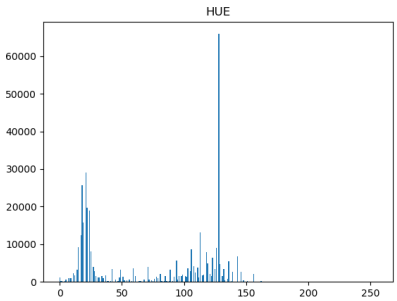


(b) GREEN

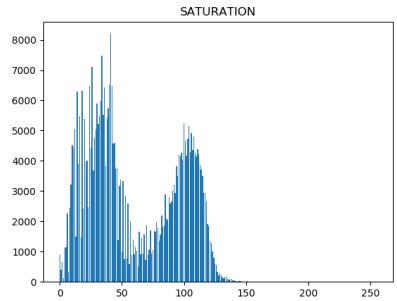


(c) BLUE

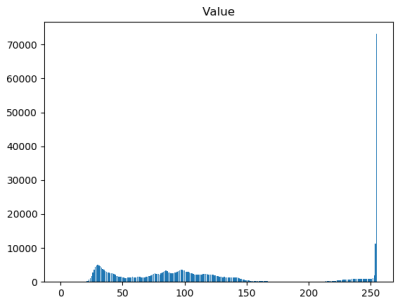
Figure 1.3: RGB Histograms



(a) HUE



(b) SATURATION



(c) VALUE

Figure 1.4: HSV Histograms

1.3 Conclusion of Task 1

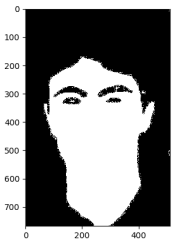
In this task, I have recognized that in different color spaces, different axes can be used for feature extraction for future applications. In RGB space, Green and Blue channels look like similar, on the other hand in HSV space Value is similar to Green and Blue, but Hue and Saturation are totally different, we can see the differences by looking at intensity images of these channels also.

Chapter 2

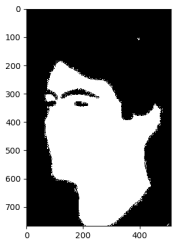
Task 2

2.1 Obtaining Binary Masks

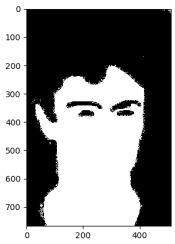
For this section, I have used ground truth images to obtain binary pixel masks. Let say, for a ground truth image with dimension $768 \times 512 \times 3$, I created an array with dimension 768×512 with full of zeros first, then I have set all the places to 255 that their corresponding place in the ground truth image is not $(0, 0, 0)$.



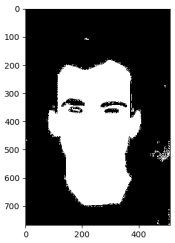
(a) image 001



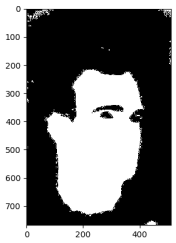
(b) image 002



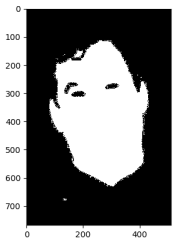
(c) image 003



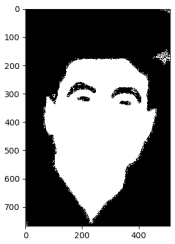
(d) image 004



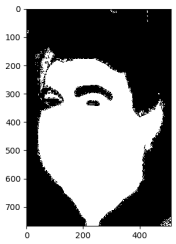
(e) image 005



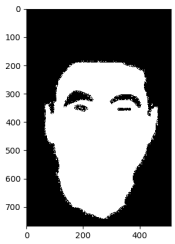
(f) image 006



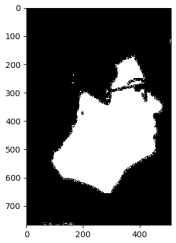
(g) image 007



(h) image 008



(i) image 009

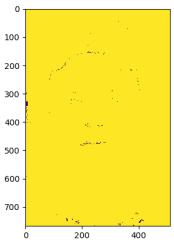


(j) image 010

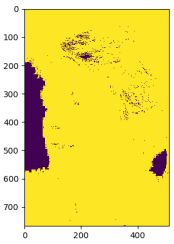
Figure 2.1: Binary Masks

2.2 Skin Color Segmentation

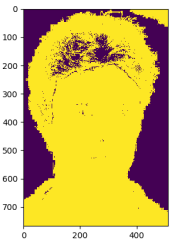
In this part, I have determine a color range for skins by using binary masks that I have obtained in the first part. And in similar way to first part, I have created skin color masks.



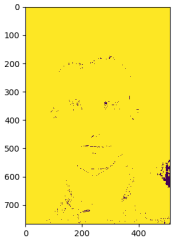
(a) image 001



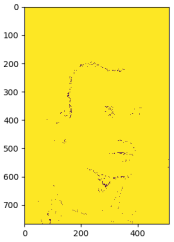
(b) image 002



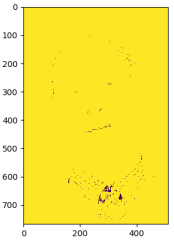
(c) image 003



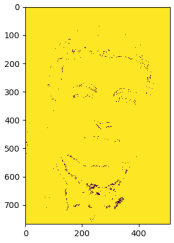
(d) image 004



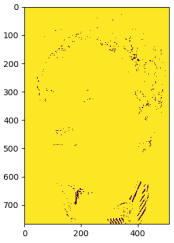
(e) image 005



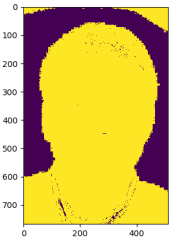
(f) image 006



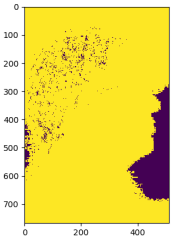
(g) image 007



(h) image 008



(i) image 009



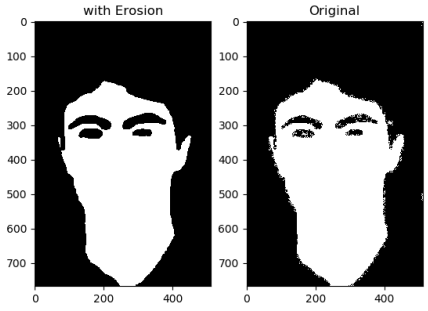
(j) image 010

Figure 2.2: Skin Color Masks

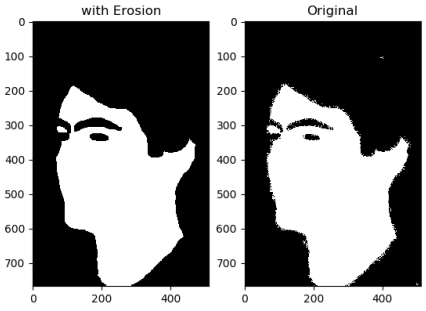
As you can easily observe that, by using those binary masks, obtained ranges are working very bad. So in part 3, I have tried morphological operation "erosion" to get better masks, hence better skin color segmentation results.

2.3 Skin Color Segmentation after Erosion

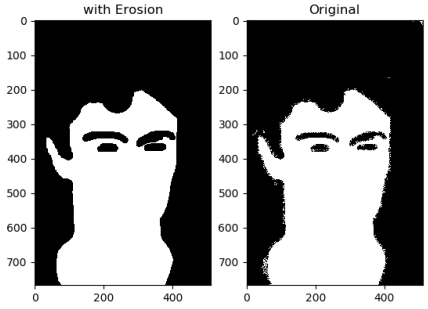
For the last part of this task, I have eroded binary masks that I have obtained in first step. Then, repeated the skin color segmentation procedure. So, for binary masks results follows:



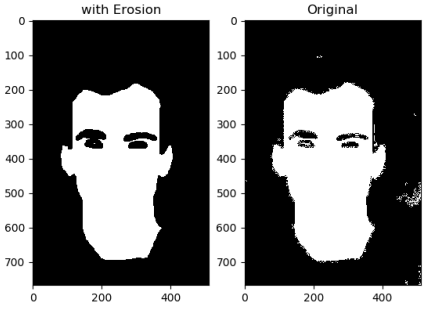
(a) image 001



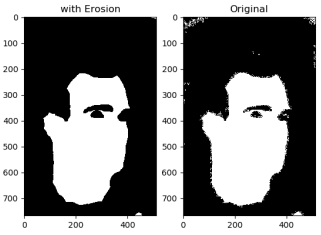
(b) image 002



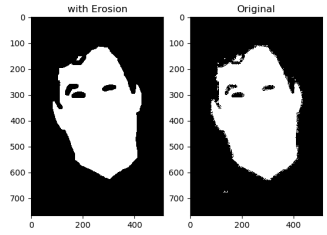
(c) image 003



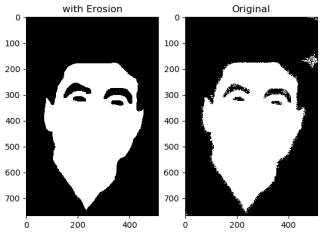
(d) image 004



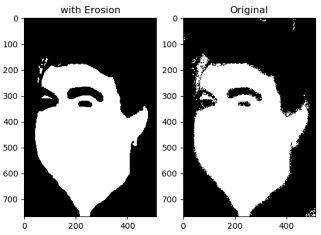
(e) image 005



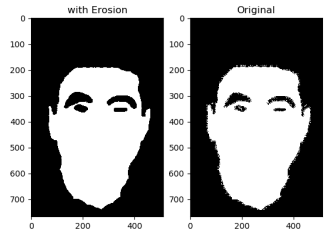
(f) image 006



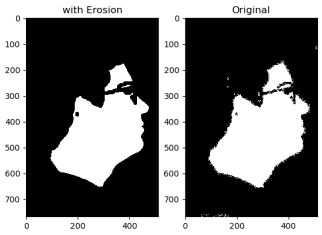
(g) image 007



(h) image 008



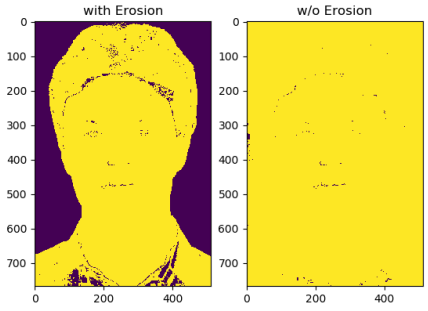
(i) image 009



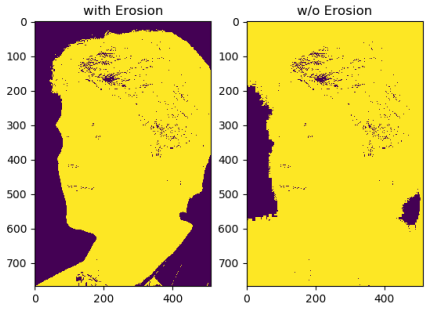
(j) image 010

Figure 2.3: Eroded Binary Masks

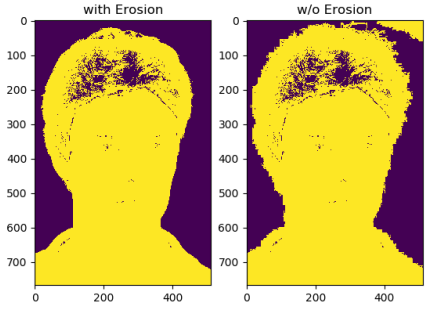
And resulting skin color masks follow:



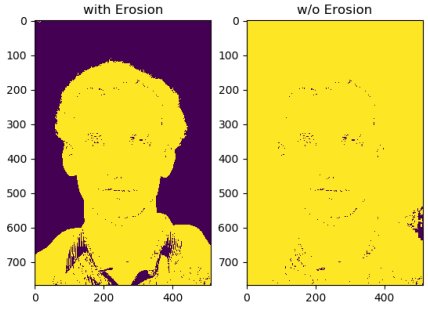
(a) image 001



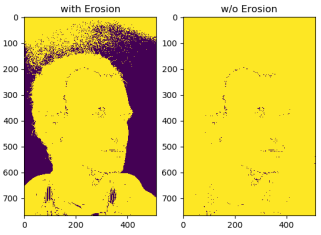
(b) image 002



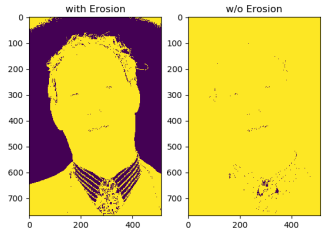
(c) image 003



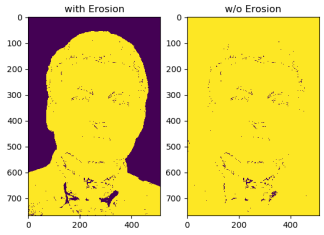
(d) image 004



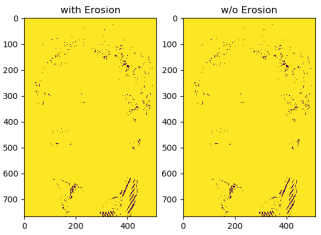
(e) image 005



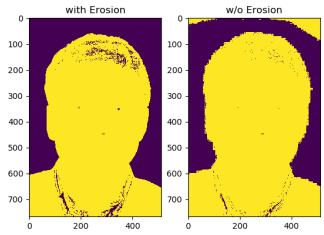
(f) image 006



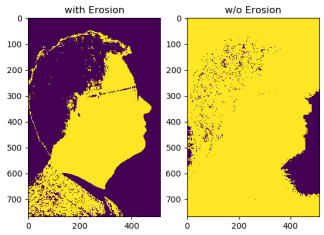
(g) image 007



(h) image 008



(i) image 009



(j) image 010

Figure 2.4: Skin Color Masks with-w/o Erosion

2.4 Conclusion

For this part, I have realized that morphological operations might be very useful to eliminate noises from the image in such operations.

Chapter 3

Task 3

3.1 K-Means Algorithm

K-Means is an unsupervised clustering algorithm. For this assignment, I have implemented K-Means similar to Matrix Factorization technique. For detailed explanation of this, you can see the GitHub notes from A.Taylan Cemgil. I got center update rule of my k-means class from his notes.

3.2 Obtaining Skin Clusters

3.2.1 First Attempt

My first attempt to obtain skin clusters was to plot k-means clustering results for each image for k from 2 to 10. Let see the results,

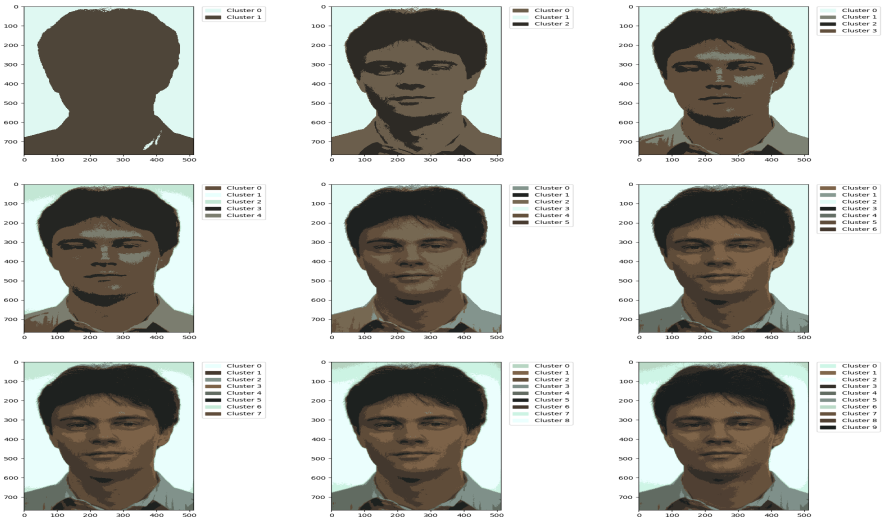


Figure 3.1: Clustering Results for image 001

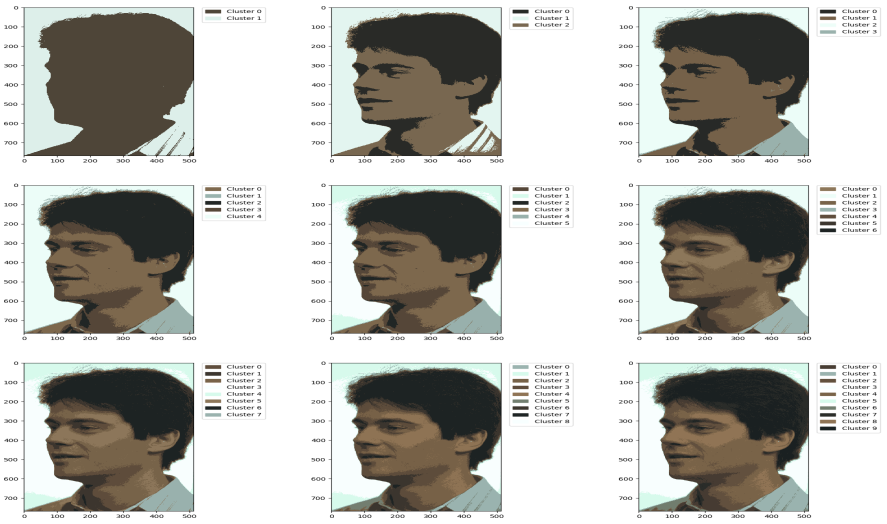


Figure 3.2: Clustering Results for image 002

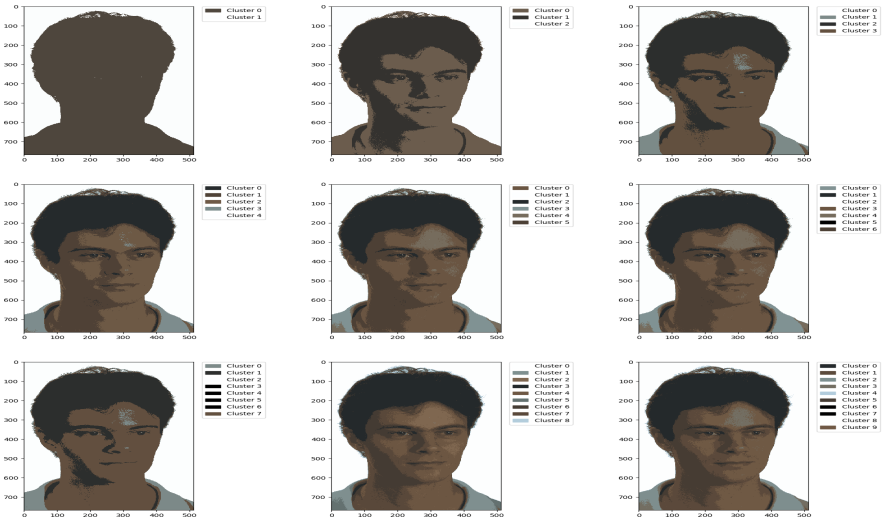


Figure 3.3: Clustering Results for image 003

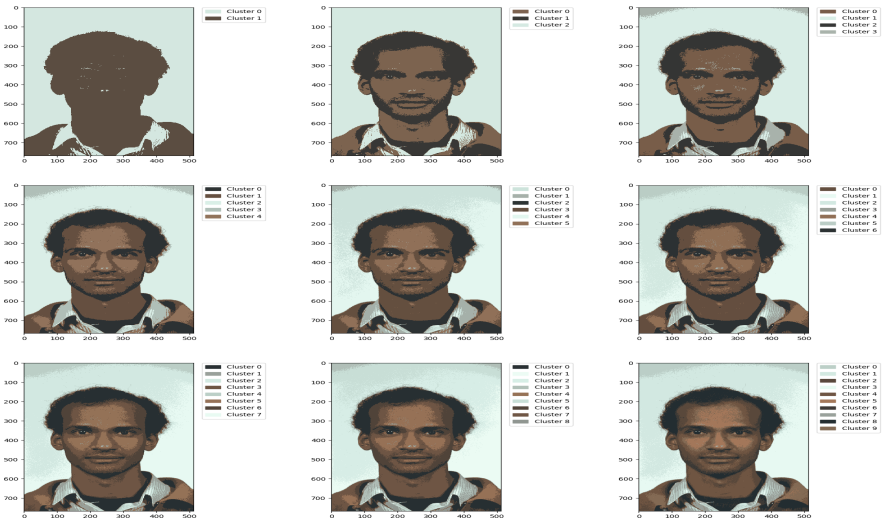


Figure 3.4: Clustering Results for image 004

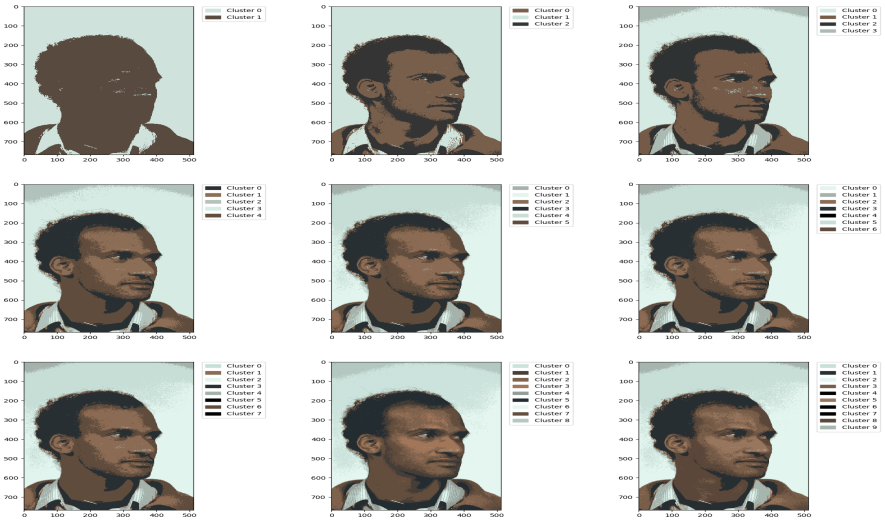


Figure 3.5: Clustering Results for image 005

Figure 3.6: Clustering Results for image 006

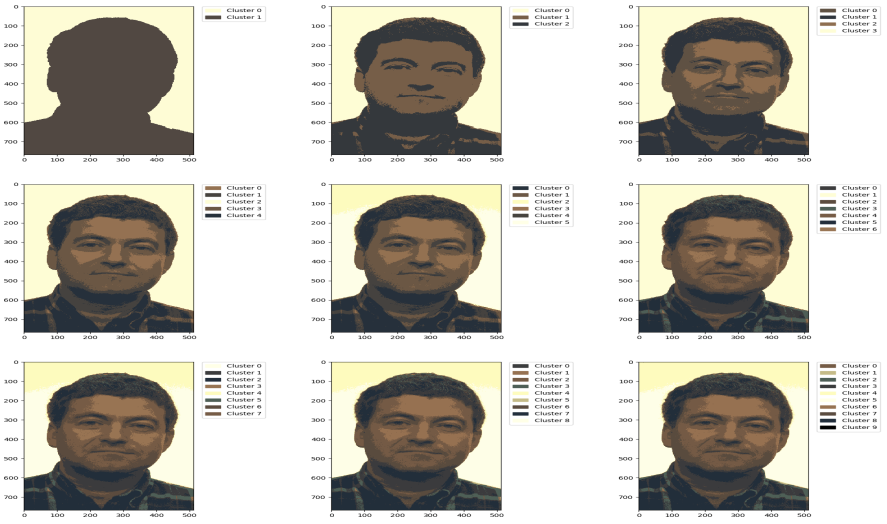


Figure 3.7: Clustering Results for image 007

Figure 3.8: Clustering Results for image 008

Figure 3.9: Clustering Results for image 009

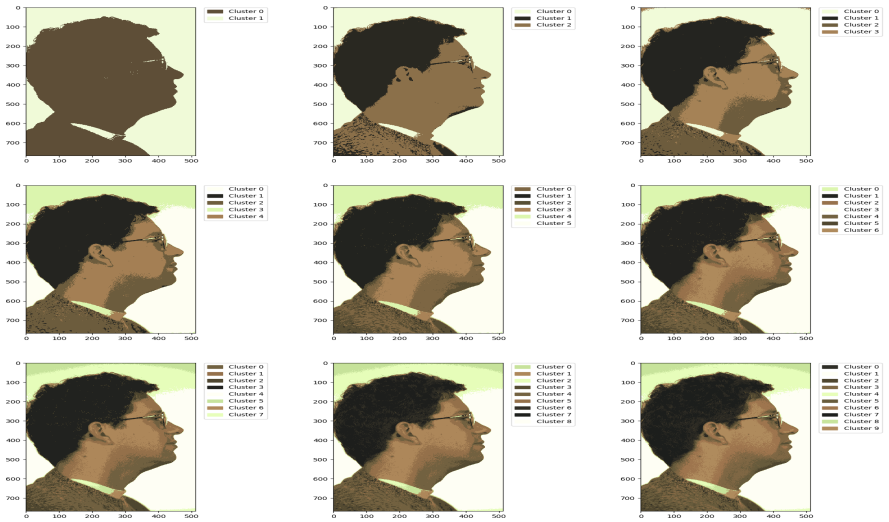


Figure 3.10: Clustering Results for image 010

Figure 3.11: Clustering Results for image 011

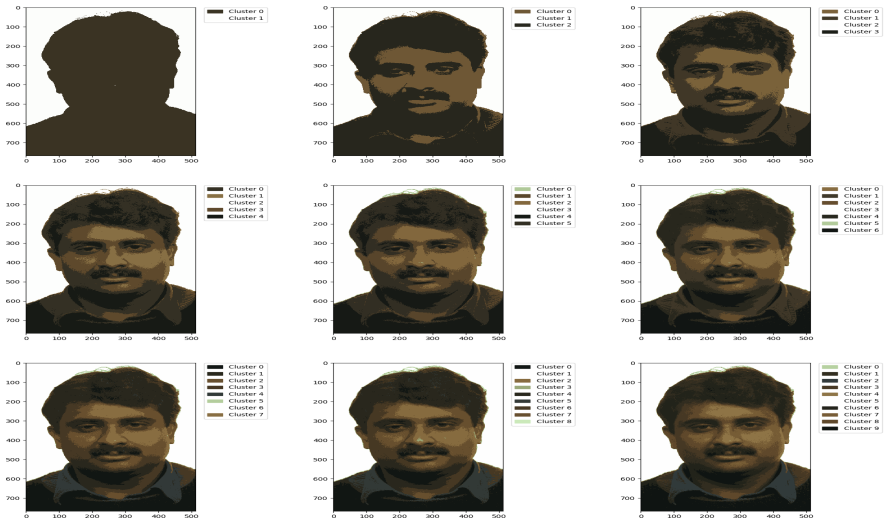


Figure 3.12: Clustering Results for image 012

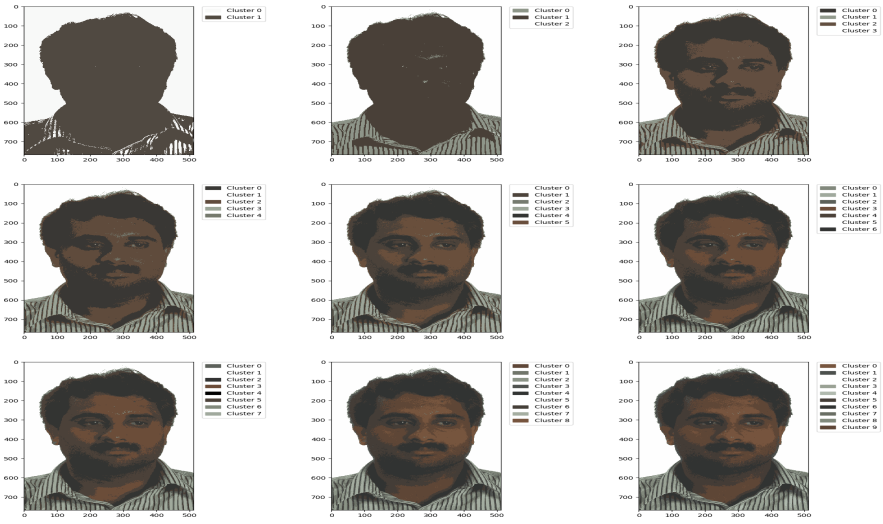


Figure 3.13: Clustering Results for image 013

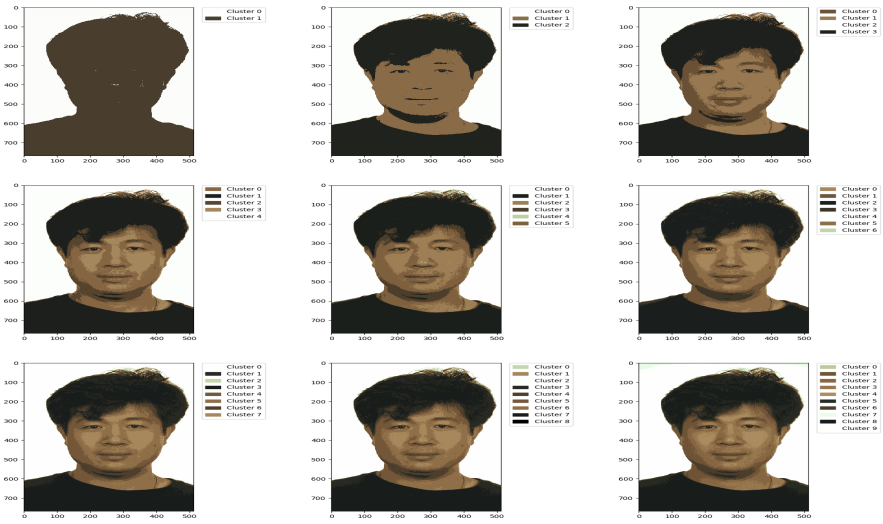


Figure 3.14: Clustering Results for image 014

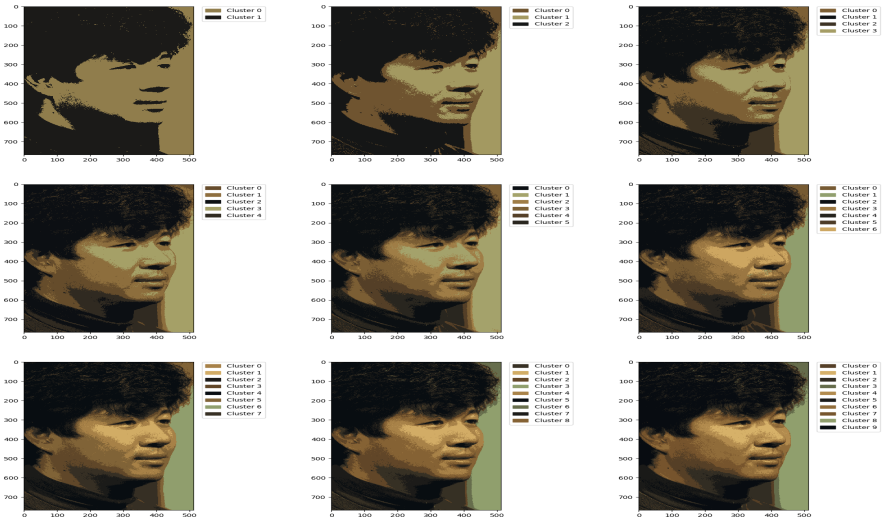


Figure 3.15: Clustering Results for image 015

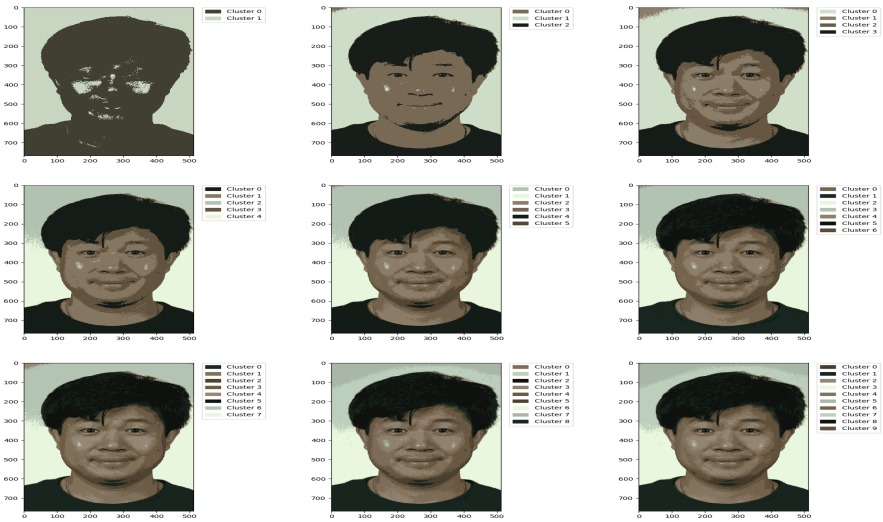


Figure 3.16: Clustering Results for image 016

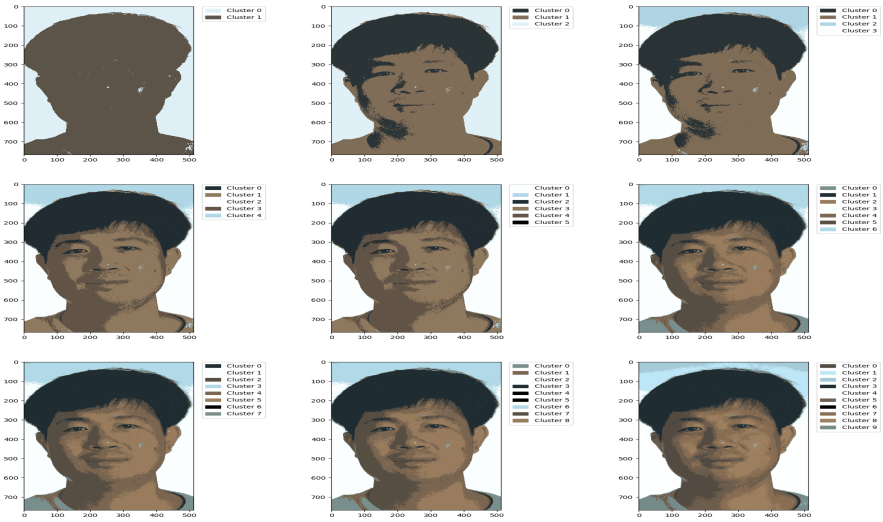


Figure 3.17: Clustering Results for image 017

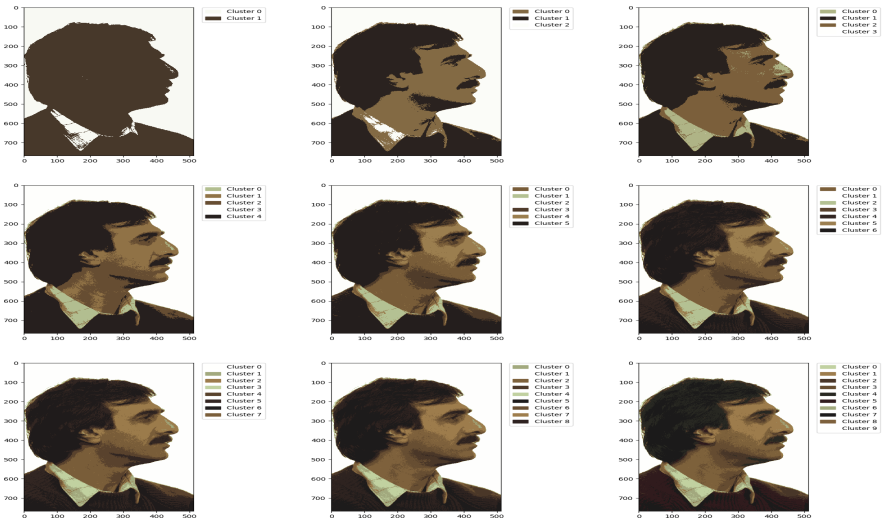


Figure 3.18: Clustering Results for image 018

Figure 3.19: Clustering Results for image 019

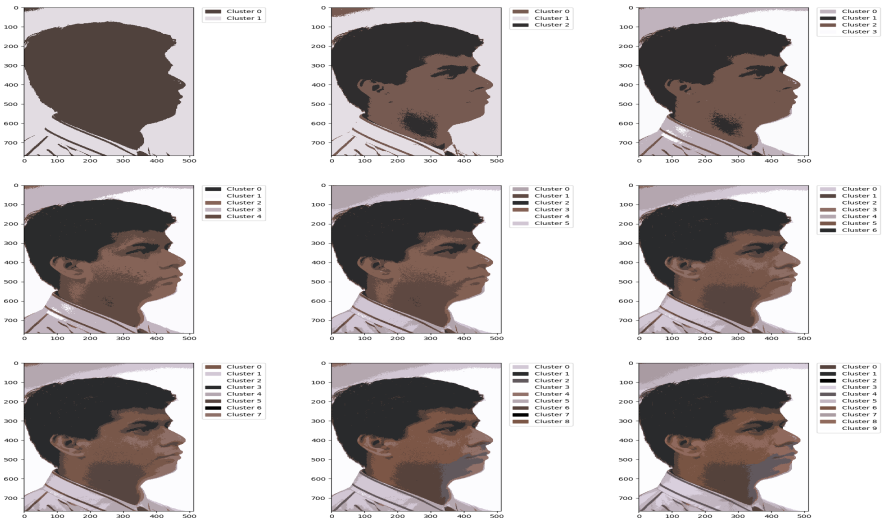
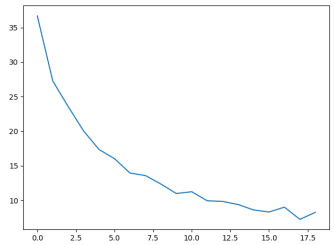


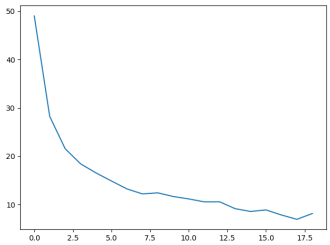
Figure 3.20: Clustering Results for image 020

After I plotted these images, I tried to select clusters that contains only skin pixels. But, it cannot be determined 100% true. My second attempt to obtain these clusters was "Elbow Rule"

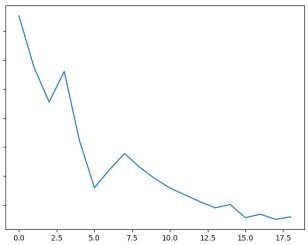
3.2.2 Second Attempt: Elbow Rule



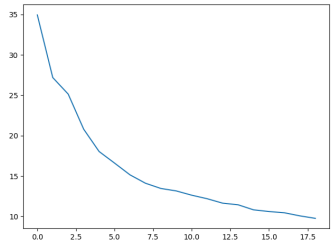
(a) Elbow Rule for img 001



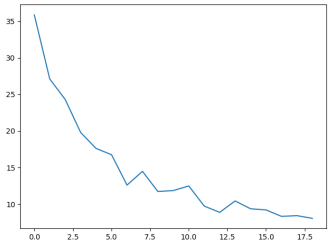
(b) Elbow Rule for img 002



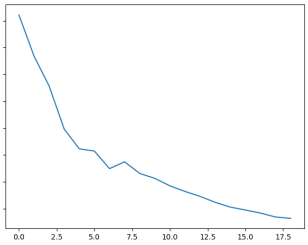
(c) Elbow Rule for img 003



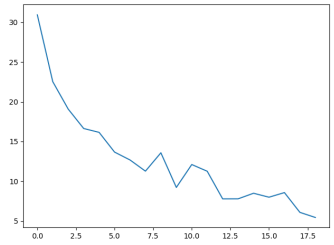
(d) Elbow Rule for img 004



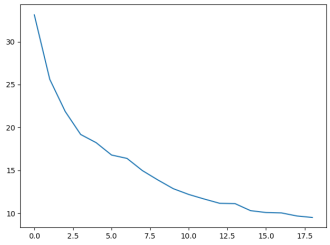
(e) Elbow Rule for img 005



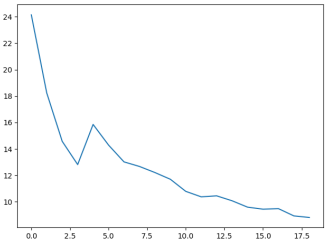
(f) Elbow Rule for img 006



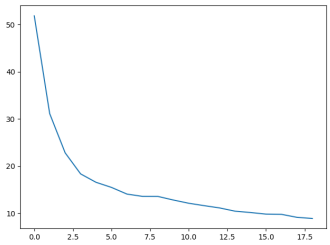
(g) Elbow Rule for img 007



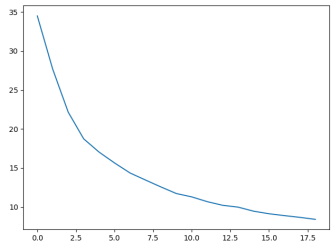
(h) Elbow Rule for img 008



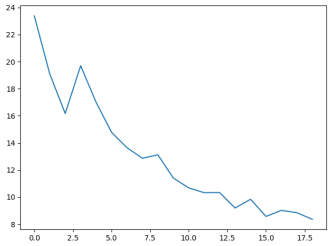
(i) Elbow Rule for img 009



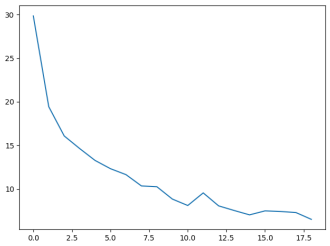
(j) Elbow Rule for img 010



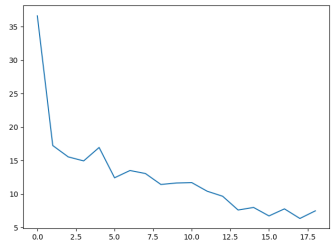
(k) Elbow Rule for img 011



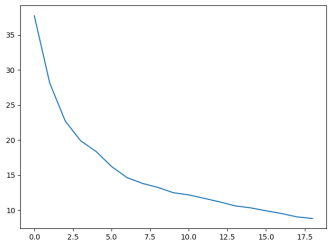
(l) Elbow Rule for img 012



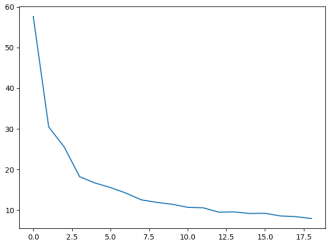
(m) Elbow Rule for img 013



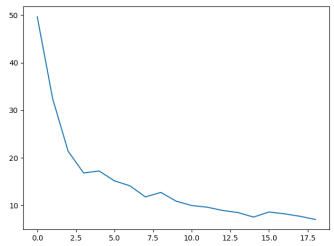
(n) Elbow Rule for img 014



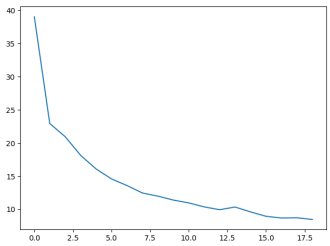
(o) Elbow Rule for img 015



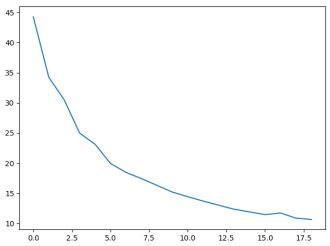
(p) Elbow Rule for img 016



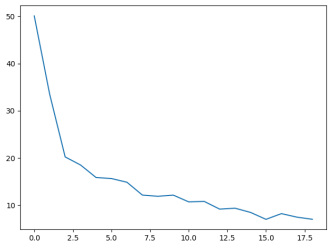
(q) Elbow Rule for img 017



(r) Elbow Rule for img 018



(s) Elbow Rule for img 019



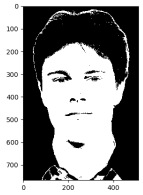
(t) Elbow Rule for img 020

Figure 3.21: Elbow Rule Graphs

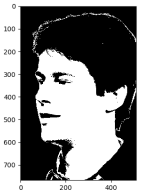
After I plotted each of these graphs, I obtained cluster ids to obtain skin pixel masks. To get same cluster ids for same colors, I set numpy random seed as 58 just for fun.

3.2.3 Results

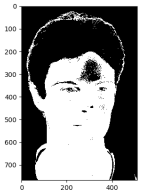
After trying 2 different approaches, I finally determined cluster ids for skin colors and obtained skin pixel masks for each image



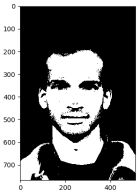
(a) Mask for image 001



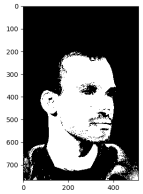
(b) Mask for image 002



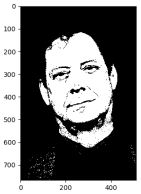
(c) Mask for image 003



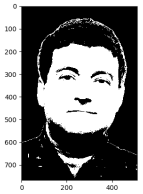
(d) Mask for image 004



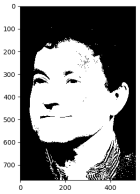
(e) Mask for image 005



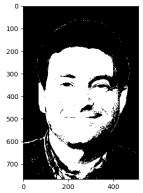
(f) Mask for image 006



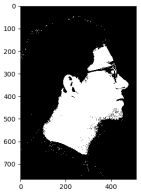
(g) Mask for image 007



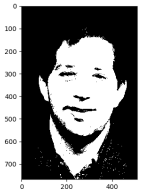
(h) Mask for image 008



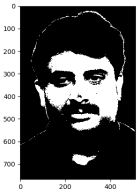
(i) Mask for image 009



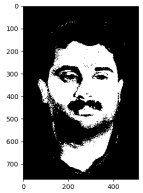
(j) Mask for image 010



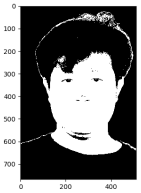
(k) Mask for image 011



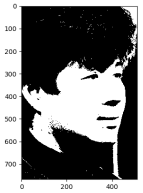
(l) Mask for image 012



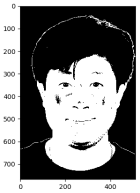
(m) Mask for image 013



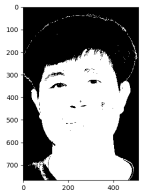
(n) Mask for image 014



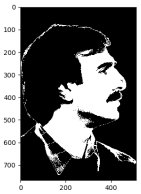
(o) Mask for image 015



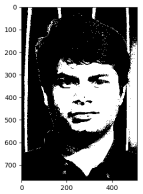
(p) Mask for image 016



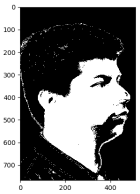
(q) Mask for image 017



(r) Mask for image 018



(s) Mask for image 019

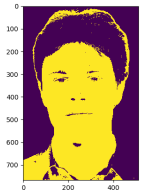


(t) Mask for image 020

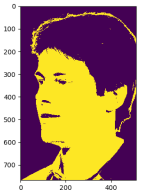
Figure 3.22: Results of obtained skin clusters

3.3 Obtain Skin Color Mask by Skin Color Segmentation

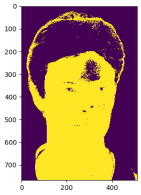
In this part, I have used skin color segmentation technique from task 2 to obtain following results,



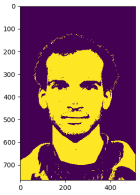
(a) Skin Color Mask for image 001



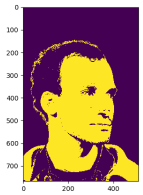
(b) Skin Color Mask for image 002



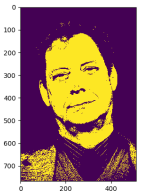
(c) Skin Color Mask for image 003



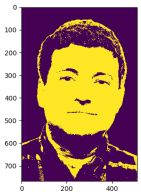
(d) Skin Color Mask for image 004



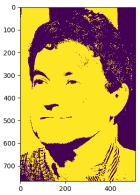
(e) Skin Color Mask for image 005



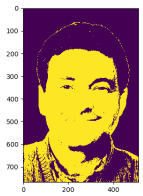
(f) Skin Color Mask for image 006



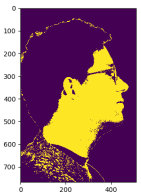
(g) Skin Color Mask for image 007



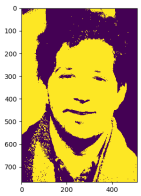
(h) Skin Color Mask for image 008



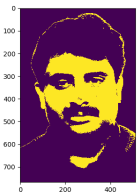
(i) Skin Color Mask for image 009



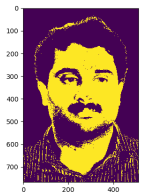
(j) Skin Color Mask for image 010



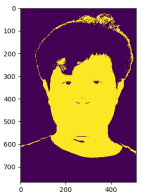
(k) Skin Color Mask for image 011



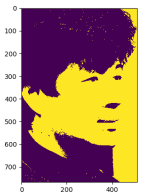
(l) Skin Color Mask for image 012



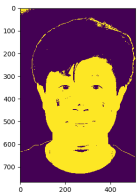
(m) Skin Color Mask for image 013



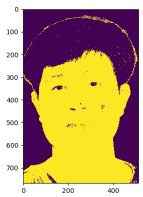
(n) Skin Color Mask for image 014



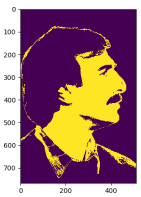
(o) Skin Color Mask for image 015



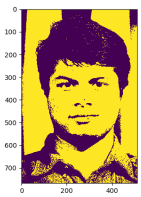
(p) Skin Color Mask for image 016



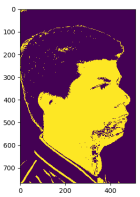
(q) Skin Color Mask for image 017



(r) Skin Color Mask for image 018



(s) Skin Color Mask for image 019



(t) Skin Color Mask for image 020

Figure 3.23: Skin Color Masks Obtained by Binary Mask results of k-Means Clustering

3.4 Results and Future Improvements

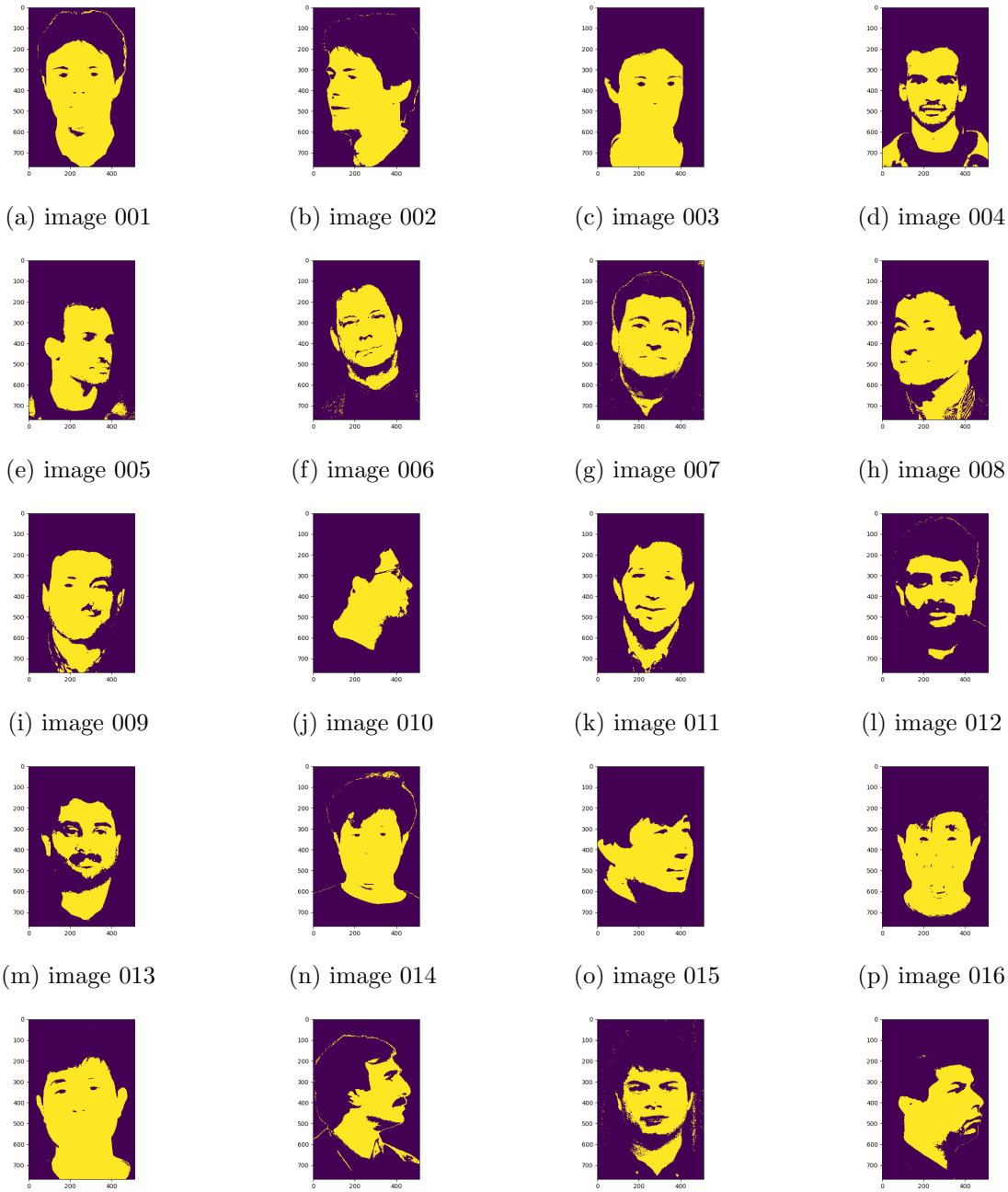
As everybody can obtain, using min-max range values to detect skin colors are not working well. To improve results of skin color segmentation, we can apply several techniques. First thing that came up is to increase cluster numbers in k-Means clustering so every skin pixel can separately obtained, but this is computationally heavy.

The second thing is to improve segmentation part. For this, we can think our skin color segmentation mechanism as a classifier which classifies between skin colors and non-skin colors. Using min-max as a classifier is a very naive approach. We can do following things;

- ➔ First, by using binary masks, we can constructs data set such that each skin pixel corresponds to 1, and non-skin pixel corresponds to 0 label
- ➔ Second, we should split data set in 2 parts train and label
- ➔ Then, we can train a binary classifier with train data set
- ➔ Test it with test data set
- ➔ Apply classifier to image colors
- ➔ Based on classifier results we can obtain a skin color mask for the image

3.4.1 Results

Let see the segmentation results from logistic regression classifier,



(a) image 001 (b) image 002 (c) image 003 (d) image 004

(e) image 005 (f) image 006 (g) image 007 (h) image 008

(i) image 009 (j) image 010 (k) image 011 (l) image 012

(m) image 013 (n) image 014 (o) image 015 (p) image 016

(q) image 017 (r) image 018 (s) image 019 (t) image 020

Figure 3.24: Logistic Regression Results

Chapter 4

References

- Logistic Regression Approach to Skin Color Segmentation
- k-Means notes from A.Taylan Cemgil GitHub page