CENG 466 Fundamentals of Image Processing Take-Home Exam 3

Mustafa Sezgin
Computer Engineering
e2380863@ceng.metu.edu.tr

I. Face Detection

A. Implementation

The tasks in the first question are implemented using `sklearn.cluster`, `skimage.color`, and `skimage.measure` packages of Scikit library.

The images are clustered with 4 color groups using `KMeans` class of `sklearn.cluster` package. The clustered color that is closest to the skin color and that contains the faces is manually selected. See Figures 1, 2 and 3.

Fig. 1. The selected clustered color for the image "1_source.png"

Fig. 2. The selected clustered color for the image "2_source.png"

After that, for the selected cluster color, the image is split into its connected regions using `label` and `regionprops` functions of `skimage.measure` package. The regions that contain faces are selected manually, and the bounding boxes are drawn. See Figures 4 and 5.

Finally, original images with the bounding boxes drawn on them are returned as outputs.

Fig. 3. The selected clustered color for the image "3_source.png"

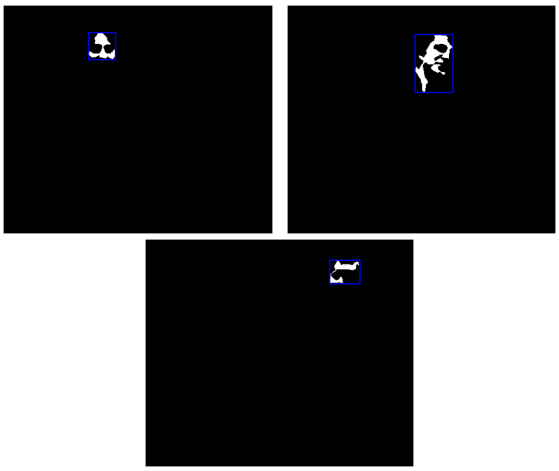

Fig. 4. The face regions of the image "1_source.png"

B. Drawbacks

The drawbacks of the algorithm used in this question are as follows.

- The algorithm requires manual selection of the skin color.
- The algorithm requires manual selection of the regions that contain faces because the largest regions on these images are on the background.
- The algorithm relies on a non-deterministic K-means clustering algorithm because it starts with randomly selected pixels.

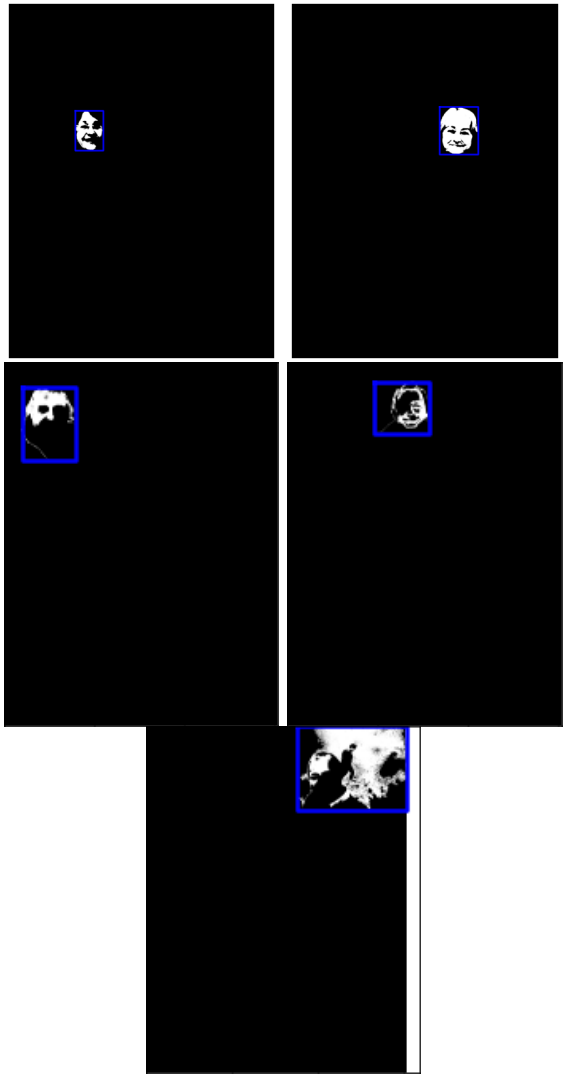

Fig. 5. The face regions of the image "2_source.png"

II. Pseudo-Coloring

A. Definition

Pseudo-coloring is a technique used to display monochrome images in color. It involves assigning different colors to different intensity levels in the image, typically using a color map or look-up table. The resulting image appears to have more than one color, although it is still composed of only one channel of data.

Pseudo-coloring is often used in image processing and computer vision to highlight certain features or patterns in the image, or to create an enhanced visual representation of the data. It can also be used to convert grayscale images to color for display or printing purposes.

There are several different approaches to pseudo-coloring an image, depending on the specific application and the desired result. Some common techniques include color mapping, color scaling, and color grading.

B. Implementation

The tasks in the second question are implemented using the `rgb2gray` and `rgb2hsv` functions from `skimage.color` package of Scikit library.

The implemented algorithm works by creating a color map from gray scale to RGB colors using the gray scaled version of the source image. To do that, first, the RGB source image is converted to gray image. Then, the average of red, green, and blue values that are converted to the same gray value is stored in an array of size 256×3 for the mapping. Finally, all the gray values of the target gray image are replaced with the colors according to the mapping. A pseudo-code for the algorithm is as follows.

```
# read the gray image
gray_img = read_image("1.png", rgb=False)

# read the RGB source image
rgb_img = read_image("1_source.png")

# convert the rgb image to gray image
rgb_gray_img = rgb2gray(rgb_img)

# create a color map from gray to color
# take average of R, G and B values of
# the pixels converted to the same gray level
color_map = np.empty((256, 3))
for l in range(256):
    pixels_l = rgb_img[rgb_gray_img == l]
    color_map[l] = np.mean(pixels_l, axis=0)

# create the pseudo-colored rgb image
colored_img = np.empty((m, n, 3))
for l in range(256):
    colored_img[gray_img == l] = color_map[l]

# write pseudo-colored image
write_image(colored_img, "1_colored.png")
```

Fig. 6. Pseudo-code for the pseudo-coloring algorithm

The sample outputs of this algorithm can be seen in Figures 7, 8, 9, and 10.

Fig. 7. "1_colored.png"

C. Drawbacks

The drawbacks of the algorithm used in this question are as follows.

Fig. 8. "2_colored.png"

Fig. 9. "3_colored.png"

- The images are not colorful as wanted.
- The algorithm could not color "4.png"; on the contrary, the output is pure black. See Figure 10.

D. Alternative Implementation

An alternative algorithm is implemented for pseudo-coloring. In this case, the algorithm works by creating a color map from gray scale to RGB colors using the intensity value of the HSV version of the source image. To do that, first, the RGB source image is converted to HSV. Then, the average of hue and saturation values that have the same intensity value is stored in an array of size 256×2 for the mapping. Finally, all the gray values of the target gray image are replaced with the colors according to the mapping.

Fig. 10. "4_colored.png"

The sample outputs of this algorithm can be seen in Figures 11, 12, 13, and 14.

Fig. 11. "1_colored_alternative.png"

Fig. 12. "2_colored_alternative.png"

Fig. 13. "3_colored_alternative.png"

E. Drawbacks

The drawbacks of the alternative algorithm used in this question are as follows.

- The images are colorful now, but look random.
- The alternative algorithm could not color "4.png" either; however, the output is not pure black but the same as the original gray image. See Figure 14.

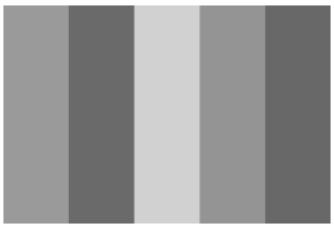

Fig. 14. "4_colored_alternative.png"

III. Plot RGB and HSV Channels

Plot of the red, green, and blue channels, and hue, saturation, and value channels of pseudo-colored images can be seen in Figures 15-22

Fig. 15. Red, green, and blue channels of "1_colored.png"

Fig. 16. Hue, saturation, and value channels of "1_colored.png"

Fig. 17. Red, green, and blue channels of "2_colored.png"

IV. Edge Detection

Edge detection algorithm is applied

- separately on red, green, and blue channels
- only on the intensity channel

of images. The result are the same for both implementations. See Figures 23, 24, 25, and 26.

Fig. 18. Hue, saturation, and value channels of "2_colored.png"

Fig. 19. Red, green, and blue channels of "3_colored.png"

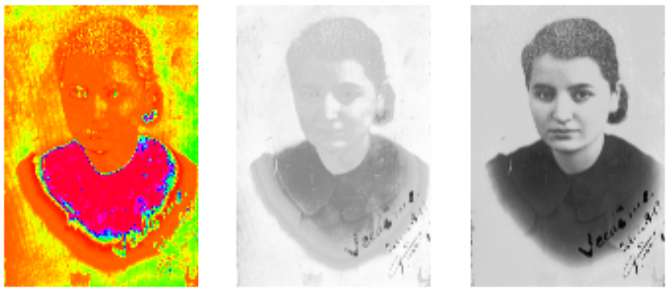

Fig. 20. Hue, saturation, and value channels of "3_colored.png"

Fig. 21. Red, green, and blue channels of "4_colored.png"

Fig. 22. Hue, saturation, and value channels of "4_colored.png"

Fig. 23. “1_colored_edges.png”

Fig. 24. “2_colored_edges.png” (Edges can barely be seen)

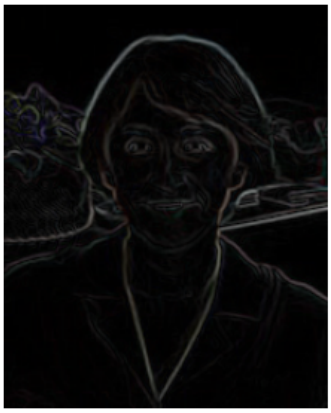

Fig. 25. “3_colored_edges.png”

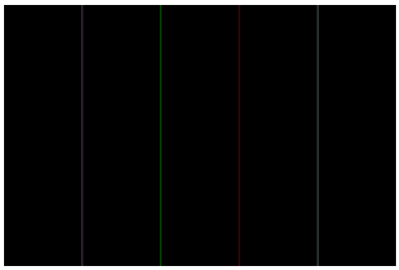

Fig. 26. “4_colored_edges.png”