PROJECT

Course: EE495 Special Topics in Computer Engineering

Image Segmentations

List of authors

Haneen Alamoudi	1708436
Marya Bamakhramah	1706689
Raghad Tariq	1707114
Reem Alsubaie	1706725

Advisor

Dr. Hemalatha Mahalingam

Department of Electrical and Computer Engineering

Faculty of Engineering

King Abdulaziz University Jeddah – Saudi Arabia

SPRING 2021

Table of Contents

Objectives

- Apply the theoretical concepts learnt in the lectures.
- Implementing coding methods learnt in the course.
- Perform a case study about image segmentation.
- Gain better understanding related to the image segmentation.
- Provide an image segmentation based on colors and shapes.

Case study description

Image segmentation is a crucial step in the image processing applications. It is a technique for dividing a digital image into groups of pixels. It is a burgeoning field of study, with applications ranging from machine vision to medical imagery to traffic and surveillance. Python provides a powerful library called "skimage" that has a wide number of image processing algorithms. This project will implement two methods of segmentation, color-based image segmentation and active contour segmentation.

In the first application, a set of sample images will be used to implements the color-based image segmentation. It will segment the images to 6 sets of colors: pink, yellow, red, green, white and blue. In addition, it will print a scatter plot of the color gradations of the image based on the LAB colors, where “L” represents the lightness, “A” represents the colors from green to red and “B” represents the colors from blue to yellow.

In the active contour segmentation, a circle will be plotted on the targeted area of the image, which is the face of the person. Then, inside the circle, the algorithm will segment the face of the person from the rest of an image by fitting a closed curve to the edges of the face.

Image Segmentations benefits and users.

The purpose of segmentation is to simplify and change the representation of an image toward something more meaningful and easier to analyze. Image segmentation is typically used to locate the objects and boundaries, such as lines and curves in images. More accurately, image segmentation is the process of assigning a label to every pixel in an image. Thus, each pixel with the same label will share a specific characteristic. Moreover, each pixel in a region will share similar characteristics or computed properties, such as color, intensity, or texture. Therefore, many users will find this method beneficial which is as follows:

- Medical Section
    - Virtual surgery simulation
    - Surgery planning
    - Measure tissue volume
    - Detecting cancerous cells
- Robotics
    - Object detection
    - Automatic Target Recognition

- Phone companies
    - Face recognition
    - Fingerprint recognition
- Video surveillance systems
    - Pedestrian detection
    - Brake light detection
- Traffic control systems
    - Traffic Monitoring
- Computer Scientist
    - Image Compression
    - Image Restoration
    - Image Enhancement

Lastly, the goals of image processing can be summarized into five group:

1. Hallucination - monitor the objects that are not visible .
2. Image restoration and sharpening - For creating a better image.
3. Image repossession - search for the image of interest .
4. Measurement of pattern – Measures a range of objects in an image .
5. Image acknowledgment – differentiate the objects in an image.

Code 1: Color Segmentation

In this section, the code of the color segmentation will be demonstrated along with testing 3 different samples. The process followed in this code is that it first converts the imported image from RGB to CIE LAB color space. CIE LAB color space defines colors in a 3D spectrum consisting of three values: *L** for perceptual lightness, and *a** and *b** for the four unique colors of human vision: red, green, blue, and yellow, Figure 1 illustrates the CIE LAB space. After obtaining the plot of the LAB space using the scatter plot, the code will use this plot to separate the colors segments and provide the user with the final result.

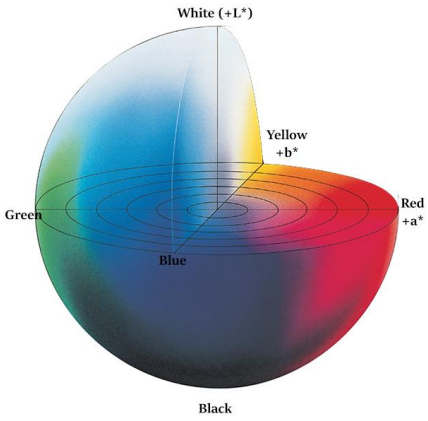

Figure 1 CIA LAB Space [6]

The final obtained color segments will be pink, yellow, red, green, white, and blue. Below is the used code for the color segmentation:

```
#needed imports
import matplotlib.pyplot as plt
from skimage.io import imread
from skimage import color
import numpy as np

#import image for segmentation, uncomment one at a time.

cimage = imread('sample1.jpg')
#cimage = imread('sample2.jpg')
#cimage = imread('sample3.jpeg')
fig, ax = plt.subplots(figsize=(20,20))
ax.imshow(cimage)
ax.axis('off')

# convert the image from RGB to CIE LAB (Lightness, red, green,
blue, and yellow).
lab_img = color.rgb2lab(cimage)
x,y,z = lab_img.shape

# for plotting the colors,the RGB values from the image will be
used.
to_plot = cimage.reshape(x*y, 3)
colors_map = to_plot.astype(np.float)/256

# create dataset for scatter plot
scatter_x = []
scatter_y = []
for xi in range(x):
    for yi in range(y):
        L_val = lab_img[xi,yi][0]
        A_val = lab_img[xi,yi][1]
        B_val = lab_img[xi,yi][2]
        scatter_x.append(A_val)
        scatter_y.append(B_val)

plt.figure(figsize=(20,20))
plt.xlabel("a* from green to red")
plt.ylabel("b* from blue to yellow")
plt.scatter(scatter_x,scatter_y, c=colors_map)

#method to filter the colors obtained from LAB plot.
#inputs in order are Lightness, red, green, blue, and yellow.
def filter_color(L_val_min, A_val_min, A_val_max, B_val_min,
B_val_max):
    filtered_image = np.copy(cimage)
    for xi in range(x):
        for yi in range(y):
            L_val = lab_img[xi,yi][0]
```

```
            A_val = lab_img[xi,yi][1]
            B_val = lab_img[xi,yi][2]
            if L_val > L_val_min and A_val > A_val_min and A_val <
A_val_max  and B_val > B_val_min and B_val < B_val_max:
                pass
            else:
                filtered_image[xi, yi] = [255,255,255]
    return filtered_image

#filtering each colors, the values may change to get better results
for each sample.
lab_img = color.rgb2lab(cimage)
yellow = filter_color(70, -50, 0, 30, 100)
red = filter_color(30, 25, 100, 0, 100)
green = filter_color(50, -128, -20, 0, 50)
blue = filter_color(50,-40, 30, -128, -20)
white = filter_color(93, -25, 25, -25, 25)
pink = filter_color(50, 20,128,-50,0)

#desply the obtained color segments.
fig, ax = plt.subplots(nrows=3, ncols=2, figsize=(20,20))
ax[0][0].imshow(pink)
ax[0][0].set_title("pink")
ax[0][0].axis('off')

ax[0][1].imshow(yellow)
ax[0][1].set_title("yellow")
ax[0][1].axis('off')

ax[1][0].imshow(red)
ax[1][0].set_title("red")
ax[1][0].axis('off')

ax[1][1].imshow(green)
ax[1][1].set_title("green")
ax[1][1].axis('off')

ax[2][0].imshow(white)
ax[2][0].set_title("white")
ax[2][0].axis('off')

ax[2][1].imshow(blue)
ax[2][1].set_title("blue")
ax[2][1].axis('off')
```

Output:

Sample 1:

Figure 2 shows the image used for the first sample, this image contains many colors and the output must have all colors segments. Figure 3 illustrates the scatter plot of the LAB colors for this sample.

Figure 2 First Sample

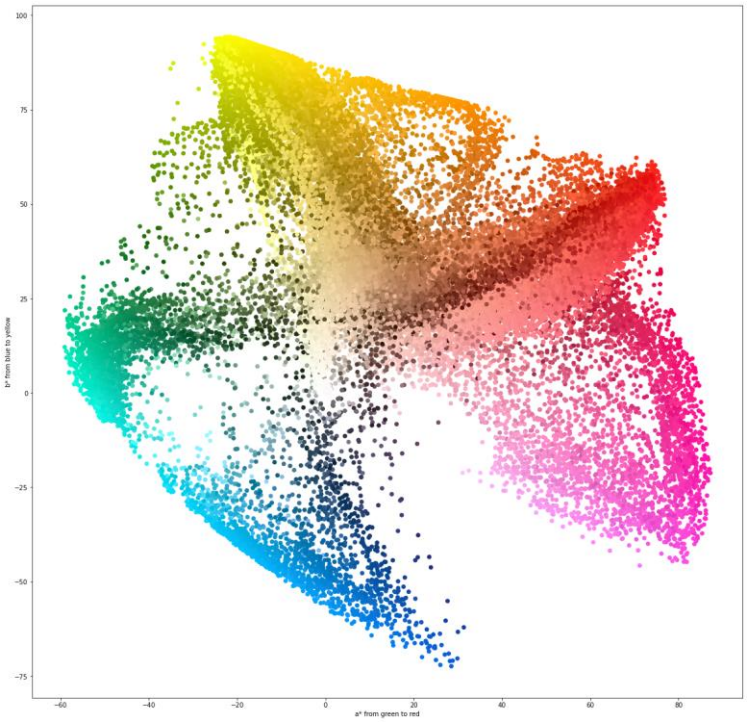

Figure 3 Scatter Plot of LAB Colors of Sample 1

The segmentation output can be seen in figure 4. where the candies were divided by colors.

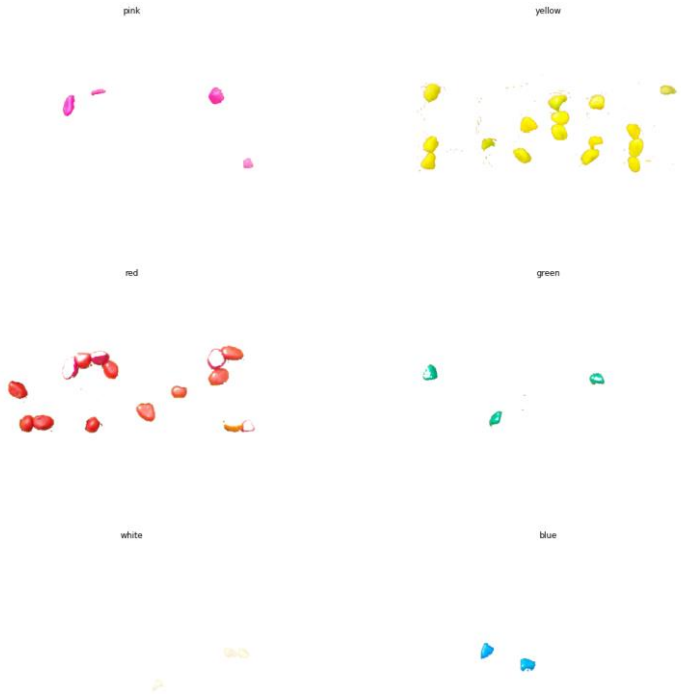

Figure 4 Output of Sample 1

Sample 2:

Figure 5 shows the image used as sample 2, this image contains only green/yellow and red. Figure 6 shows its scatter plot while figure 7 shows the output, where the segments were only yellow, green, and red. This sample size is smaller than sample 1 and the segmentation were obtained faster, thus, larger images take more time for the segmentation to be completed.

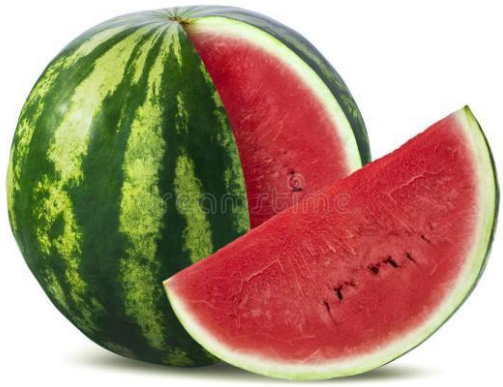

Figure 5 Second Sample

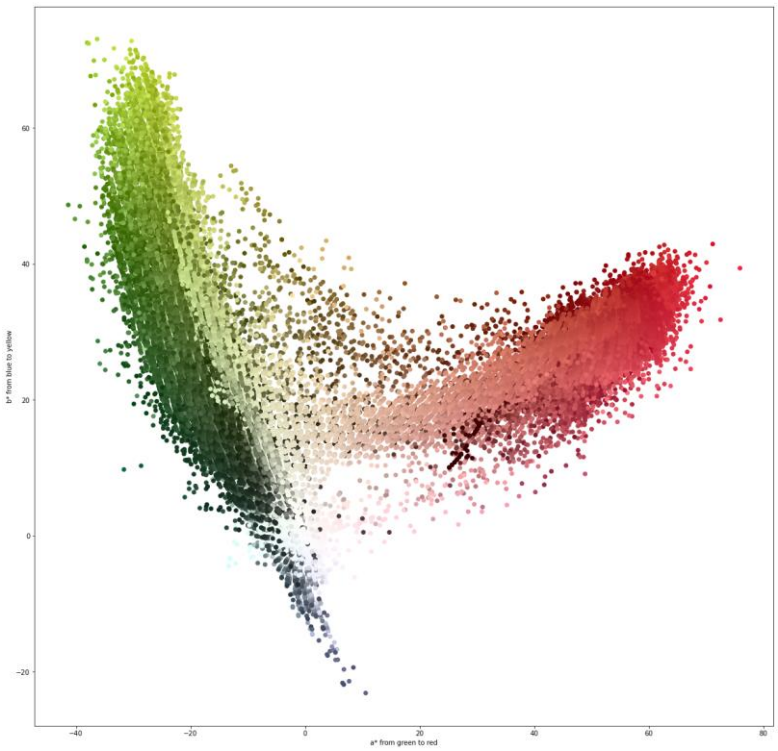

Figure 6 Scatter Plot of LAB Colors of Sample 2

Figure 7 Output of Sample 2

Sample 3:

Figure 8 contains the image of sample 3, this sample contains only pink, red, yellow, and green. Figure 9 shows the scatter plot and figure 10 shows the output.

Figure 8 Third Sample

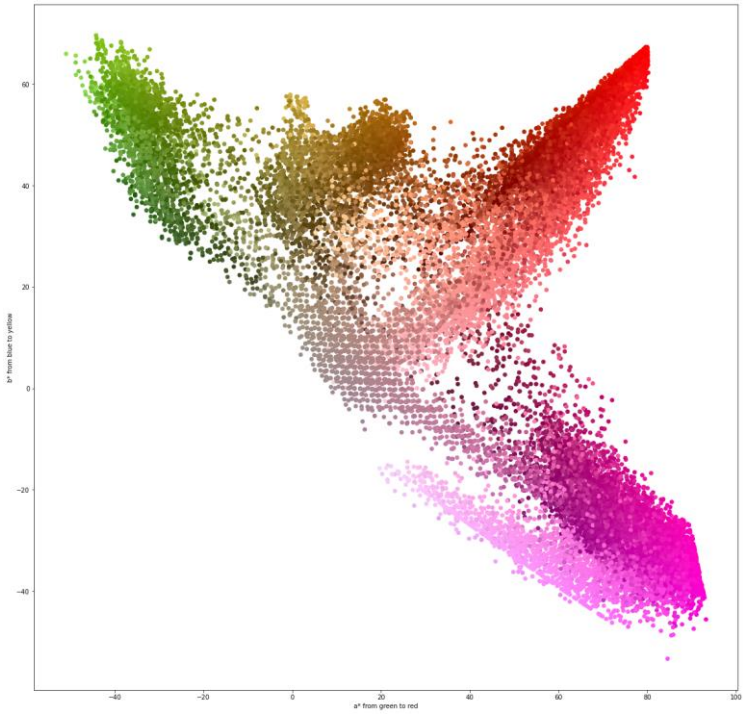

Figure 9 Scatter Plot of LAB Colors of Sample 3

Figure 10 Output of Sample 3

In all three samples the background was ignored by the code, this provides better color segmentation and can be used for object detection.

Code 2: Supervised and Unsupervised Segmentations

Here we discuss supervised and unsupervised segmentation, supervised is done through the use of active contour segmentation called snakes and the unsupervised uses Simple Linear Iterative Clustering (SLIC). Using high contrast image for segmentation will need advanced tools so in this code we convert the image into grayscale to increase the image noise.

Supervised: Active contour segmentation

Active Contour segmentation also called snakes. It works by using a user defined contour surrounding the area of interest as an input and the contour then slowly contracts and is attracted or repelled from light and edges.

Two tests were done, first an example shown in Figure 11 (C) that segments the face of a person from the rest of an image by fitting a closed curve to the edges of the face using the default parameter. After that the alpha and beta values were manipulated to find a better fit, increasing the value of alpha will produce a faster contract and increasing value of beta will make the result smoother, the result after manipulation is shown in Figure 11 (D)

The second test is done using a picture of an air balloon and the segmentation is implemented to segments the balloon from the rest of the image as shown in Figure 12 (C). Moreover, further tweaking is done to the beta and alpha values to achieve a better result that is shown in Figure 12 (D).

```
import numpy as np
import matplotlib.pyplot as plt
import skimage.data as data
import skimage.segmentation as seg
import skimage.filters as filters
import skimage.draw as draw
import skimage.color as color
# import the image
from skimage import io

def image_show(image, nrows=1, ncols=1, cmap='gray'):
    fig, ax = plt.subplots(nrows=nrows, ncols=ncols, figsize=(14,
14))
    ax.imshow(image, cmap='gray')
    ax.axis('off')
    return fig, ax

"""
TEST 1: Girl image
```

```
"""
image = io.imread('girl.jpg')
plt.imshow(image);

# gray image plot
image_gray = color.rgb2gray(image)
image_show(image_gray);

# A function to find the points that make a circle using center and
radius
def circle_points(resolution, center, radius):
    """
    Generate points which define a circle on an image.Centre refers
to the centre of the circle
    """
    radians = np.linspace(0, 2*np.pi, resolution)
    c = center[1] + radius*np.cos(radians)#polar co-ordinates
    r = center[0] + radius*np.sin(radians)

    return np.array([c, r]).T
# Exclude last point because a closed path should not have duplicate
points
points = circle_points(200, [110, 310], 100)[:-1]

# image plot with the circle points
fig, ax = image_show(image)
ax.plot(points[:, 0], points[:, 1], '--r', lw=3)

# image plot with the circle and the contour segmentation with
dafault values
snake = seg.active_contour(image_gray, points)
fig, ax = image_show(image)
ax.plot(points[:, 0], points[:, 1], '--r', lw=3)
ax.plot(snake[:, 0], snake[:, 1], '-b', lw=3);

# image plot with the circle and the contour segmentation with
alpha=0.06,beta=0.3
snake = seg.active_contour(image_gray, points,alpha=0.06,beta=0.3)
fig, ax = image_show(image)
ax.plot(points[:, 0], points[:, 1], '--r', lw=3)
ax.plot(snake[:, 0], snake[:, 1], '-b', lw=3);
```

Test 1:

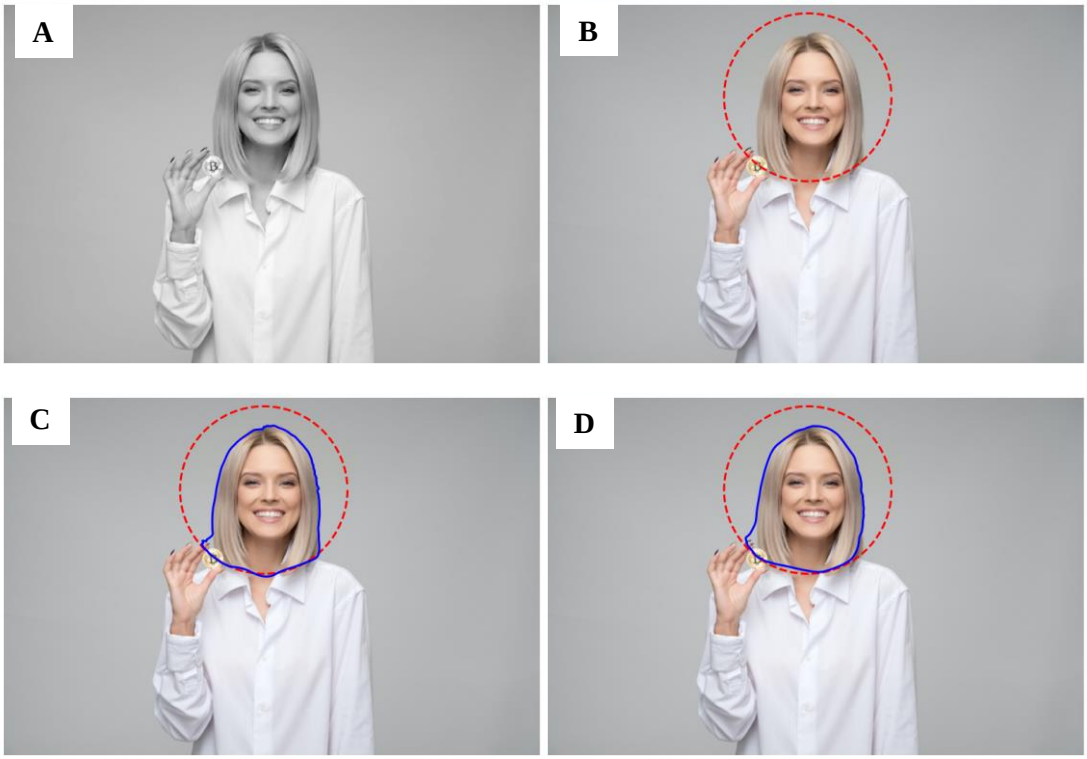

Figure 11: Supervised active contour segmentation using the girl image, (A) gray scale image, (B) the image with a circle around the region of interest, (C) segmentation result with the default values, and (D) segmentation result with the default values alpha=0.06 and beta=0.3

Test 2:

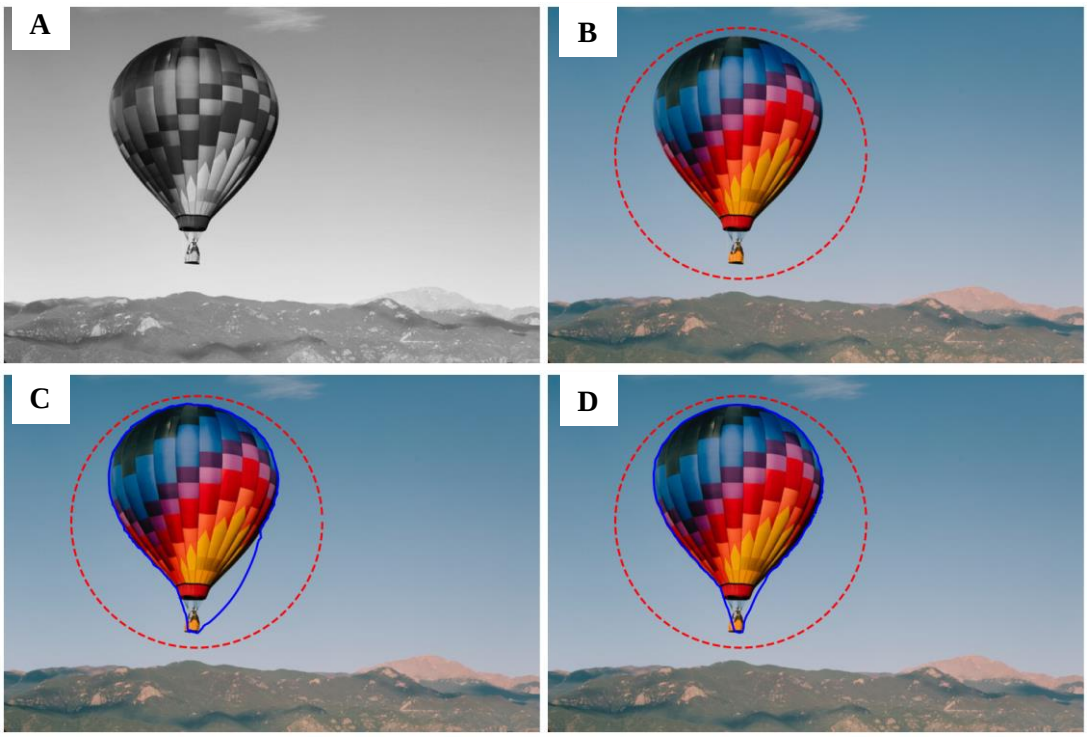

Figure 12: Supervised active contour segmentation using the air balloon image, (A) gray scale image, (B) the image with a circle around the region of interest, (C) segmentation result with the default values, and (D) segmentation result with the default values alpha=0.06 and beta=0.3

Unsupervised segmentation: SLIC (Simple Linear Iterative Clustering)

Starting with a large image, unsupervised segmentation is able to divide the image into several sub-regions reducing a large number of pixels in fewer regions. Simple Linear Iterative Clustering uses K-Means as machine-learning algorithm. It works by taking a colored image and tries to separate the pixels values into a specific number of sub-regions.

Here, two tests were done, the first is using the girl image and the number of regions is 155, and the resulted image is shown in Figure 13. The second test is shown in Figure 14 and is done with the air balloon image and a number of regions equals 200.

```
"""
TEST 2: Girl image
"""
# segments the image and sperate the pixels of the image into 155
regions
image_slic = seg.slic(image,n_segments=155)

# label2rgb replaces each discrete label with the average interior
color
image_show(color.label2rgb(image_slic, image, kind='avg'))
```

Test 1: segments the image into 155 regions

Figure 13: Unsupervised segmentation using the girl image and number of segments = 155

Test 2: segments the image into 200 regions

Figure 14: Unsupervised segmentation using the air balloon image and number of segments = 200

References

[1] A. Kharwal, “Image segmentation with python,” 22-Oct-2020. [Online]. Available: https://thecleverprogrammer.com/2020/09/01/image-segmentation-with-python/. [Accessed: 19-Apr-2021].

[4] P. Pandey, “Image segmentation using PYTHON'S scikit-image module,” 15-Nov-2020. [Online]. Available: https://towardsdatascience.com/image-segmentation-using-pythons-scikit-image-module-533a61ecc980. [Accessed: 20-Apr-2021].

[2] B. B. ., “A STUDY ON THE IMPORTANCE OF IMAGE PROCESSING AND ITS APLLICATIONS,” *International Journal of Research in Engineering and Technology*, vol. 03, no. 15, pp. 155–160, 2014.

[3] J. A. Delmerico, P. David, and J. J. Corso, “Building facade detection, segmentation, and parameter estimation for mobile robot localization and guidance,” *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011.

[5] S. Zachow, H.-C. Hege, and M. Zilske, *3D reconstruction of individual anatomy from medical image data: segmentation and geometry processing*. Berlin-Dahlem: Konrad-Zuse-Zentrum für Informationstechnik, 2007.

[6] “What is CIE 1976 Lab Color Space?,” *Konica Minolta Color, Light, and Display Measuring Instruments*, 11-Mar-2020. [Online]. Available: https://sensing.konicaminolta.asia/what-is-cie-1976-lab-color-space/. [Accessed: 20-Apr-2021].

[7] Mokrzycki, Wojciech & Tatol, Maciej. Perceptual difference in L * a * b * color space as the base for object colour identfication. (2009). 10.13140/2.1.1160.2241.