HW4-Hamed-Mohammadzadeh-9812762418

June 30, 2022

[1]:
```
from PIL import Image
import numpy as np
from numpy import asarray
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import random
import math
from numpy.linalg import inv
import matplotlib
import cv2
```

[2]:
```
# hue range = [0, 360]
# saturation range = [0, 1]
# intensity range = [0, 256]
def rgb_to_hsi(rgb):
    hsi = np.zeros(rgb.shape)
    n, m = rgb.shape[0:2]
    max_num = 0
    max_den = 0
    max_arccos = np.arccos(-1)

    for x in range(n):
        for y in range(m):
            r, g, b = rgb[x][y].astype(np.int32)
            sum_rgb = r+g+b

            numerator = (r - g + r - b)/2
            denominator = float(((r-g)**2 + (r-b)*(g-b)))**0.5
            max_num = max(max_num, numerator)
            max_den = max(max_den, denominator)

            if numerator == 0:
                theta = np.arccos(0)
            else:
                theta = np.arccos(numerator/denominator)
                theta = theta/max_arccos * 180
```

```
            if b <= g:
                hsi[x][y][0] = theta
            else:
                hsi[x][y][0] = 360 - theta

            if sum_rgb != 0:
                hsi[x][y][1] = 1-(3*min(r, g, b)/sum_rgb)
            else:
                hsi[x][y][1] = 0

            hsi[x][y][2] = sum_rgb/3

    return hsi
```

[3]:
```
def hsi_to_rgb(hsi):
    # input hsi:
    # hue range = [0, 360]
    # saturation range = [0, 1]
    # intensity range = [0, 256]

    rgb = np.zeros((hsi.shape))
    n, m = rgb.shape[0:2]

    for x in range(n):
        for y in range(m):
            h, s, i = hsi[x][y]
            r = 0
            g = 0
            b = 0

            if h == 0:
                r = i * (1 + 2 * s)
                g = i * (1 - s)
                b = i * (1 - s)

            # RG sector
            elif h > 0 and h < 120:
                b = i * (1 - s)
                r = i * (1 + (s * np.cos(math.radians(h)) / np.cos(math.
↪radians(60-h))))
                g = (3 * i) - r - b

            elif h == 120:
                r = i - i * s
                g = i + 2 * i * s
                b = i - i * s
```

```
            # GB sector
            elif h > 120 and h < 240:
                h = h - 120
                r = i * (1 - s)
                g = i * (1 + (s * np.cos(math.radians(h)) / np.cos(math.
↪radians(60-h))))
                b = (3 * i) - r - g

            elif h == 240:
                r = i - i * s
                g = i - i * s
                b = i + 2 * i * s

            # BR sector
            elif h >= 240 and h < 360:
                h = h - 240
                g = i * (1 - s)
                b = i * (1 + ((s * np.cos(math.radians(h))) / np.cos(math.
↪radians(60-h))))
                r = (3 * i) - b - g

            rgb[x][y][0] = r
            rgb[x][y][1] = g
            rgb[x][y][2] = b

    r_max = np.amax(rgb[:, :, 0])
    b_max = np.amax(rgb[:, :, 1])
    g_max = np.amax(rgb[:, :, 2])

    # scale rgb to 0-255 (computed values in rgb CAN be bigger than 255)
    #rgb[:, :, 0] = (rgb[:, :, 0]/r_max) * 255
    #rgb[:, :, 1] = (rgb[:, :, 1]/g_max) * 255
    #rgb[:, :, 2] = (rgb[:, :, 2]/b_max) * 255

    #### instead of scaling, cap it
    np.clip(rgb[:, :, 0], 0, 255, out=rgb[:, :, 0])
    np.clip(rgb[:, :, 1], 0, 255, out=rgb[:, :, 1])
    np.clip(rgb[:, :, 2], 0, 255, out=rgb[:, :, 2])


    print(r_max, b_max, g_max)
    return rgb.astype(np.uint8)
```

[4]:
```
def rgb_histogram(img_arr, title = "RGB Histogram", ignore_black=False):
    r_map, g_map, b_map = rgb_map(img_arr, ignore_black)
```

```
    clw = 2
    plt.figure(figsize=(20, 11))
    plt.plot(r_map, color='r', label = 'Red', linewidth=clw)
    plt.plot(g_map, color='g', label = 'Green', linewidth=clw)
    plt.plot(b_map, color='b', label = 'Blue', linewidth=clw)

    plt.title(title)
    plt.ylabel("Count")

    plt.show()
```

[5]:
```
def rgb_map(img_arr, ignore_black=False):
    r_map = np.zeros(256)
    g_map = np.zeros(256)
    b_map = np.zeros(256)

    if ignore_black:
        for i in img_arr:
            for j in i:
                if j[0]!=0 and j[1]!=0 and j[2]!=0:
                    r_map[j[0]] += 1
                    g_map[j[1]] += 1
                    b_map[j[2]] += 1
    else:
        for i in img_arr:
            for j in i:
                r_map[j[0]] += 1
                g_map[j[1]] += 1
                b_map[j[2]] += 1

    return r_map, g_map, b_map
```

[6]:
```
def hsi_histogram(hsi):
    plt.figure(figsize=(20, 20))
    plt.subplot(2, 2, 1)
    plt.subplot(221),plt.hist(hsi[:, :, 0].flatten(), 75, facecolor='blue'),␣
 ↪plt.title("Hue")
    plt.subplot(222),plt.hist(hsi[:, :, 1].flatten(), 75, facecolor='green'),␣
 ↪plt.title("Saturation")
    plt.subplot(223),plt.hist(hsi[:, :, 2].flatten(), 255, facecolor='red'),␣
 ↪plt.title("Intensity")
    plt.subplot(224),plt.imshow(hsi_to_rgb(hsi)), plt.title("Original Image"),␣
 ↪plt.axis('off')
```

[7]:
```
def display_hsi(hsi):
    plt.figure(figsize=(20, 20))
    plt.subplot(2, 2, 1)
```

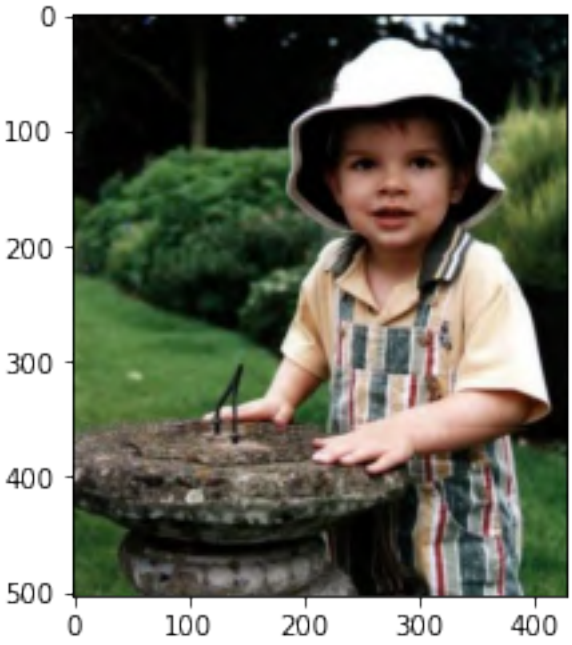
```
    plt.subplot(221),plt.imshow(hsi[:,:,0].astype('uint8'), cmap='gray'), plt.
↪title("Hue"), plt.axis('off')
    plt.subplot(222),plt.imshow(hsi[:,:,1], cmap='gray'), plt.
↪title("Saturation"), plt.axis('off')
    plt.subplot(223),plt.imshow(hsi[:,:,2].astype('uint8'), cmap='gray'), plt.
↪title("Intensity"), plt.axis('off')
    plt.subplot(224),plt.imshow(hsi_to_rgb(hsi)), plt.title("Original Image"),␣
↪plt.axis('off')
```

1 Q1

1.1 RGB image

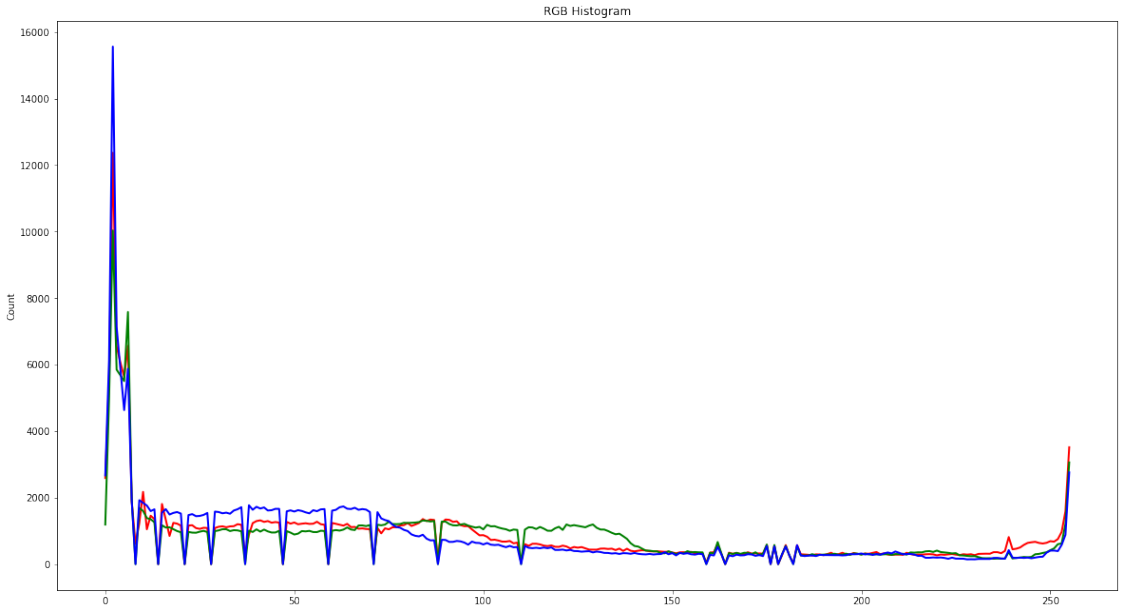
[8]:
```
rgb = cv2.imread('park.png', cv2.IMREAD_COLOR)[:,:,::-1]
print(rgb.shape)
plt.imshow(rgb)
plt.show()
```

```
(503, 428, 3)
```



1.2 RGB histogram

[9]:
```
rgb_histogram(rgb)
```

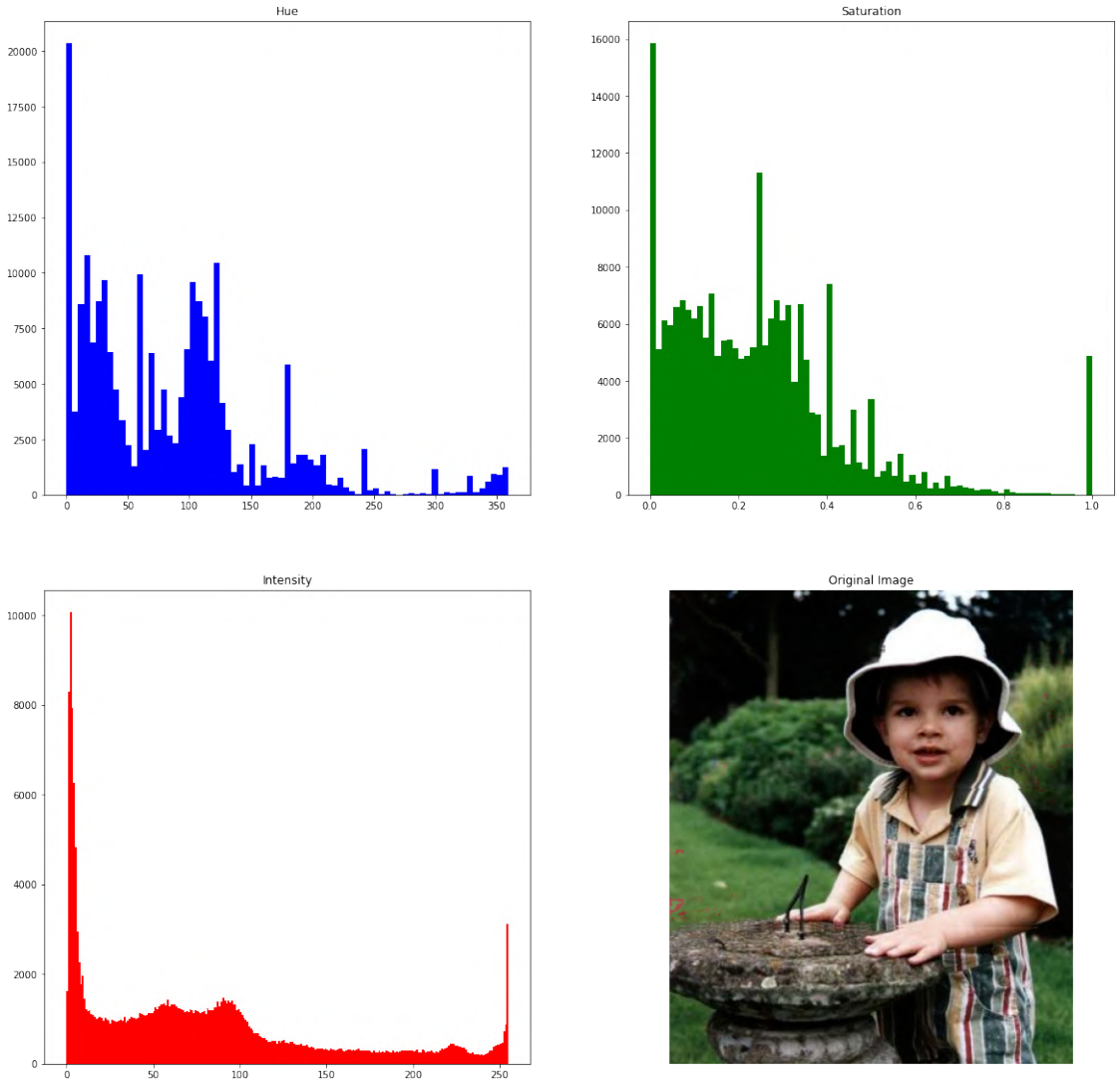


1.3 HSI conversion and HSI histogram

[10]:
```
hsi = rgb_to_hsi(rgb)
hsi_histogram(hsi)
```

```
F:\apps\Anaconda\lib\site-packages\ipykernel_launcher.py:4:
MatplotlibDeprecationWarning: Adding an axes using the same arguments as a
previous axes currently reuses the earlier instance.  In a future version, a new
instance will always be created and returned.  Meanwhile, this warning can be
suppressed, and the future behavior ensured, by passing a unique label to each
axes instance.
  after removing the cwd from sys.path.
```

```
259.3651926849884 255.00000000000003 255.00000000000003
```



1.4 Displaying HSI components

[11]:
```
display_hsi(hsi)
```

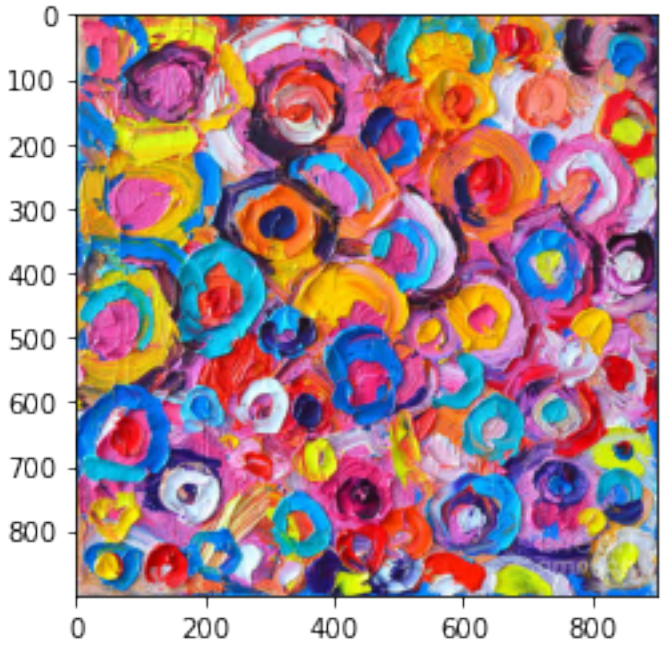
```
F:\apps\Anaconda\lib\site-packages\ipykernel_launcher.py:4:
MatplotlibDeprecationWarning: Adding an axes using the same arguments as a
previous axes currently reuses the earlier instance.  In a future version, a new
instance will always be created and returned.  Meanwhile, this warning can be
suppressed, and the future behavior ensured, by passing a unique label to each
axes instance.
  after removing the cwd from sys.path.
```

```
259.3651926849884 255.00000000000003 255.00000000000003
```

2 Q2

[12]:
```
rgb = cv2.imread('painting.jpg', cv2.IMREAD_COLOR)[:,:,::-1]
print(rgb.shape)
plt.imshow(rgb)
plt.show()
```

```
(900, 900, 3)
```

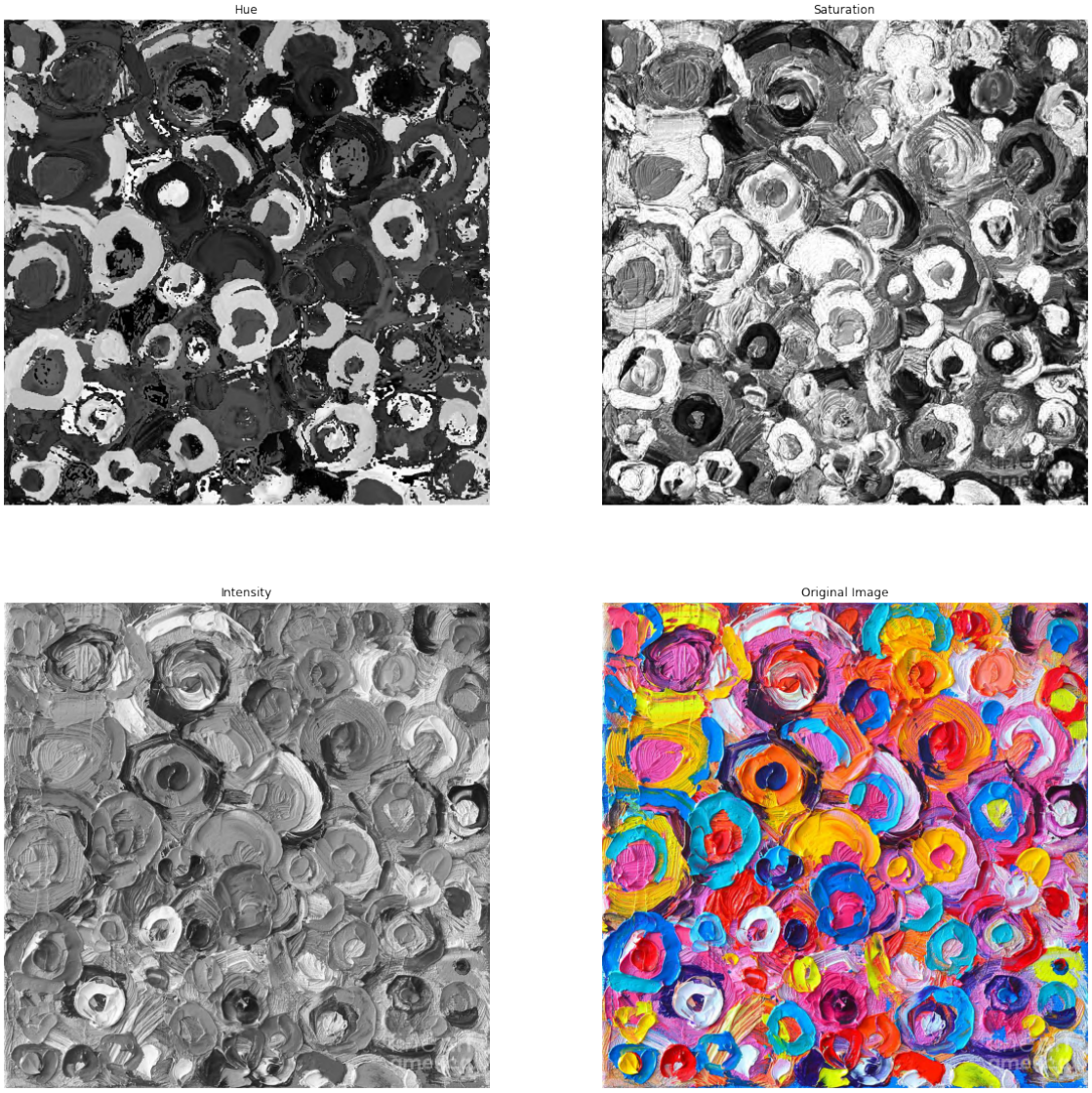


2.1 HSI conversion

[13]:
```
hsi = rgb_to_hsi(rgb)
display_hsi(hsi)
```

```
F:\apps\Anaconda\lib\site-packages\ipykernel_launcher.py:4:
MatplotlibDeprecationWarning: Adding an axes using the same arguments as a
previous axes currently reuses the earlier instance.  In a future version, a new
instance will always be created and returned.  Meanwhile, this warning can be
suppressed, and the future behavior ensured, by passing a unique label to each
axes instance.
  after removing the cwd from sys.path.
```

```
317.47299897914843 255.00000000000054 255.00000000000242
```

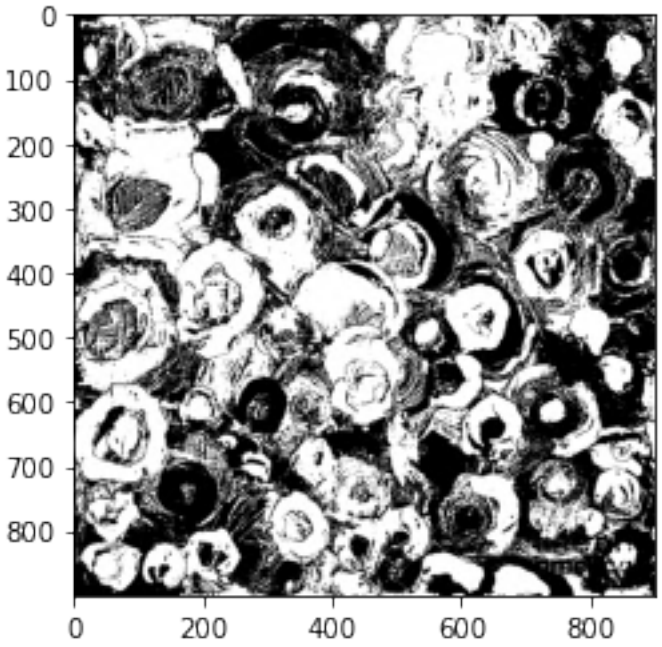


2.2 Image Segementation in HSI color space

2.2.1 Binary saturation mask

[14]:
```
# use saturation as a binary mask
sat_mask = np.zeros((hsi.shape[0:2]))

sat_mask[hsi[:, :, 1] > 0.5] = 1
sat_mask[hsi[:, :, 1] <= 0.5] = 0
plt.imshow(sat_mask, cmap='gray')
```

[14]:
```
<matplotlib.image.AxesImage at 0x25dab481d08>
```



2.2.2 Red Mask

product of Hue and Binary Saturation mask

[15]:
```
hue = hsi[:, :, 0]
red_mask = hue * sat_mask
```

2.2.3 Treshholding Red mask

based on HSI color palette, Hue in range of [330 - 30] is RED.

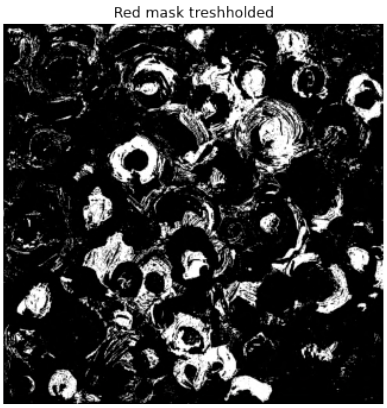
[16]:
```
# treshholding red mask
red_mask_tresh = np.zeros((red_mask.shape))
tresh1 = 330 # 0 -> 360
tresh2 = 30

if tresh1 > tresh2:
        red_mask_tresh = np.where(((red_mask >= tresh1)&(red_mask<360)) |
                                  ((red_mask <= tresh2)&(red_mask>0)), 1, 0)
else:
    red_mask_tresh = np.where((red_mask >= tresh1) & (red_mask <= tresh2), 1, 0)

#red_mask_tresh = red_mask_tresh.astype(np.uint8)
```

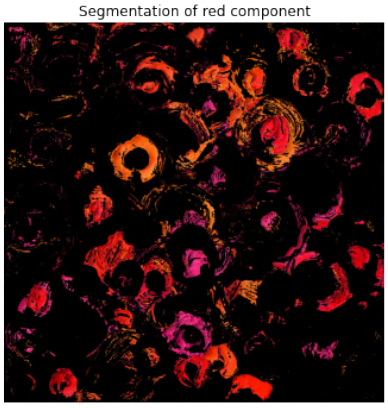
2.2.4 Segmentation of red component

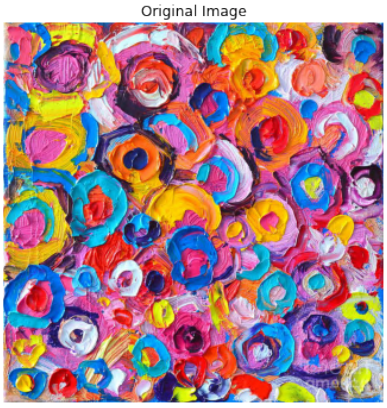
[17]:
```
red_segment = np.zeros((rgb.shape))
red_segment[:, :, 0] = rgb[:, :, 0] * red_mask_tresh
red_segment[:, :, 1] = rgb[:, :, 1] * red_mask_tresh
red_segment[:, :, 2] = rgb[:, :, 2] * red_mask_tresh


red_segment = red_segment.astype(np.uint8)

plt.figure(figsize=(20, 20))
plt.subplot(3, 2, 1)
plt.subplot(321),plt.imshow(sat_mask, cmap='gray'), plt.title("Binary␣
 ↪Saturation Mask"), plt.axis('off')
plt.subplot(322),plt.imshow(hue, cmap='gray'), plt.title("Hue"), plt.axis('off')
plt.subplot(323),plt.imshow(red_mask, cmap='gray'), plt.title("Red Mask,␣
 ↪Prodoct of Hue and Binary Saturation mask"), plt.axis('off')
plt.subplot(324),plt.imshow(red_mask_tresh, cmap='gray'), plt.title("Red mask␣
 ↪treshholded"), plt.axis('off')
plt.subplot(325),plt.imshow(red_segment), plt.title("Segmentation of red␣
 ↪component"), plt.axis('off')
plt.subplot(326),plt.imshow(rgb), plt.title("Original Image"), plt.axis('off')
```

```
F:\apps\Anaconda\lib\site-packages\ipykernel_launcher.py:11:
MatplotlibDeprecationWarning: Adding an axes using the same arguments as a
previous axes currently reuses the earlier instance.  In a future version, a new
instance will always be created and returned.  Meanwhile, this warning can be
suppressed, and the future behavior ensured, by passing a unique label to each
axes instance.
  # This is added back by InteractiveShellApp.init_path()
```

[17]:
```
(<AxesSubplot:title={'center':'Original Image'}>,
 <matplotlib.image.AxesImage at 0x25daa6f7648>,
 Text(0.5, 1.0, 'Original Image'),
 (-0.5, 899.5, 899.5, -0.5))
```

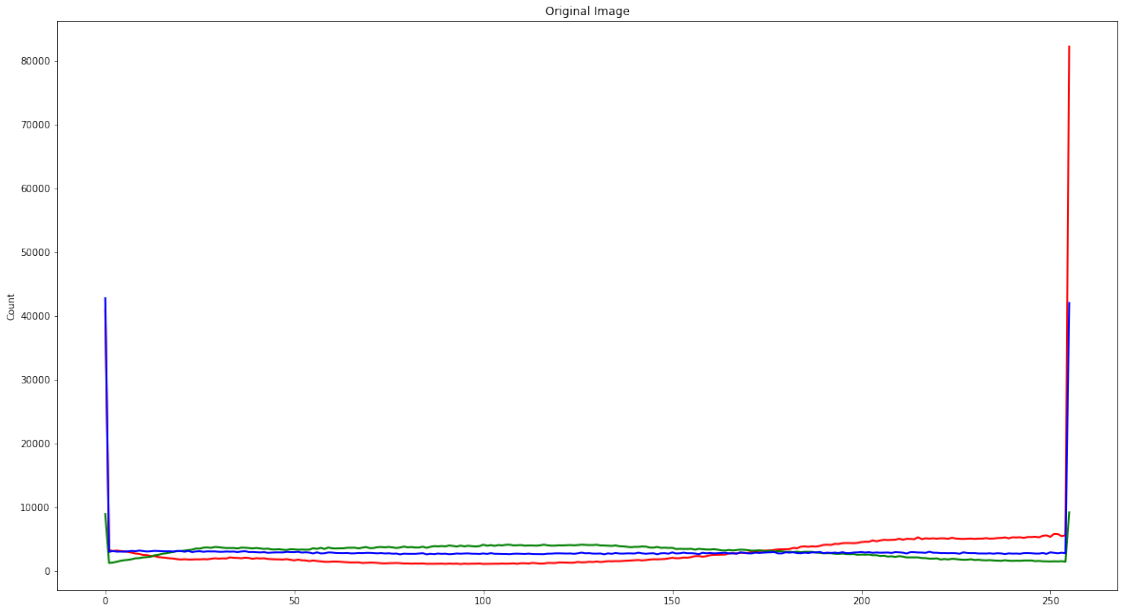


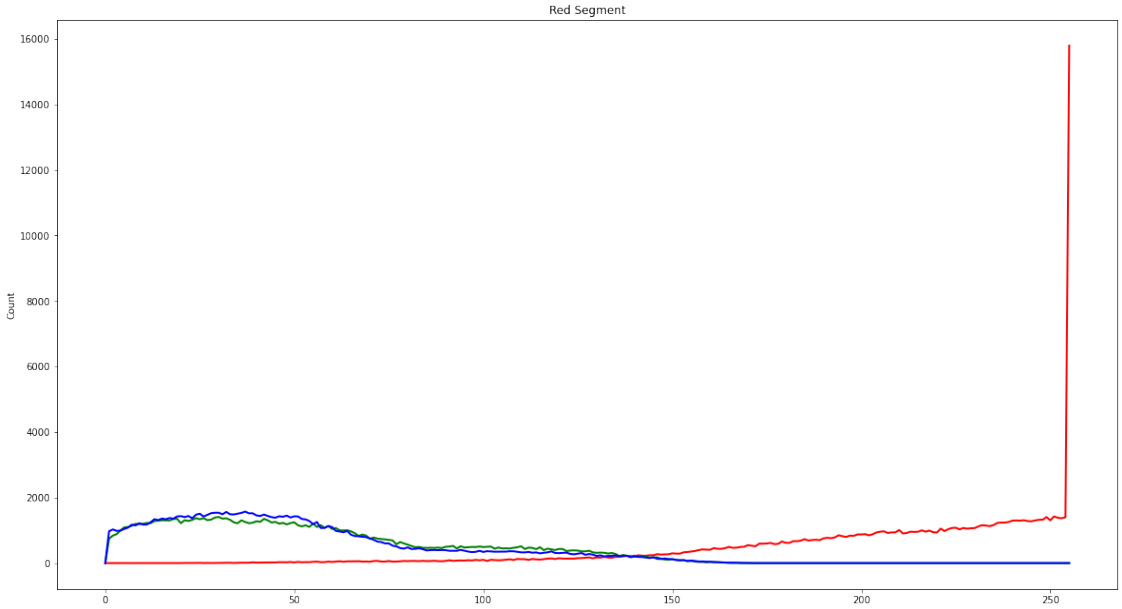




2.2.5 Comparing Histograms

[18]:
```
rgb_histogram(rgb, "Original Image"), rgb_histogram(red_segment, "Red Segment",␣
↪True)
```


Original Image
Count
80000
70000
60000
50000
40000
30000
20000
10000
0
0
50
100
150
200
250


Red Segment
Count
16000
14000
12000
10000
8000
6000
4000
2000
0
0
50
100
150
200
250

[18]: (None, None)

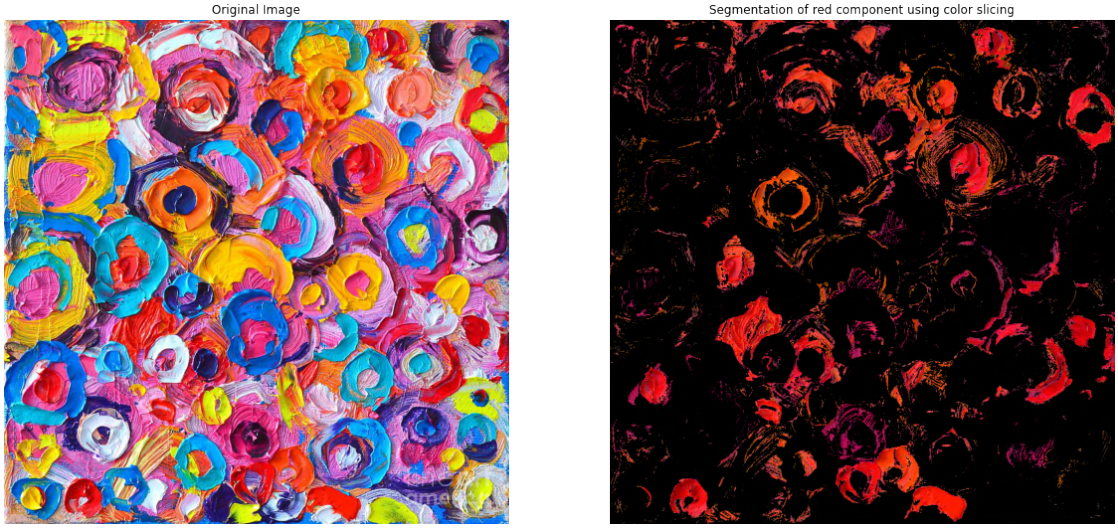
2.3 Color slicing in RGB space

[19]:
```
def rgb_color_slice(rgb_c, img, w):
    result = np.zeros((img.shape))
    n, m = (img.shape[0:2])

    for x in range(n):
        for y in range(m):
            flag = True
            for k in range(3):
                if np.abs(img[x][y][k] - rgb_c[k]) > w/2:
                    # not in the cube
                    flag = False
                    break

            if flag == True:
                # in cube
                for k in range(3):
                    result[x][y][k] = img[x][y][k]
            else:
                for k in range(3):
                    result[x][y][k] = 0

    return result
```

[20]:
```
cs = rgb_color_slice([200, 30, 30], rgb, 150).astype(np.uint8)

plt.figure(figsize=(20, 20))
plt.subplot(1, 2, 1)
plt.subplot(121),plt.imshow(rgb), plt.title("Original Image"), plt.axis('off')
plt.subplot(122),plt.imshow(cs), plt.title("Segmentation of red component using␣
 ↪color slicing"), plt.axis('off')
```

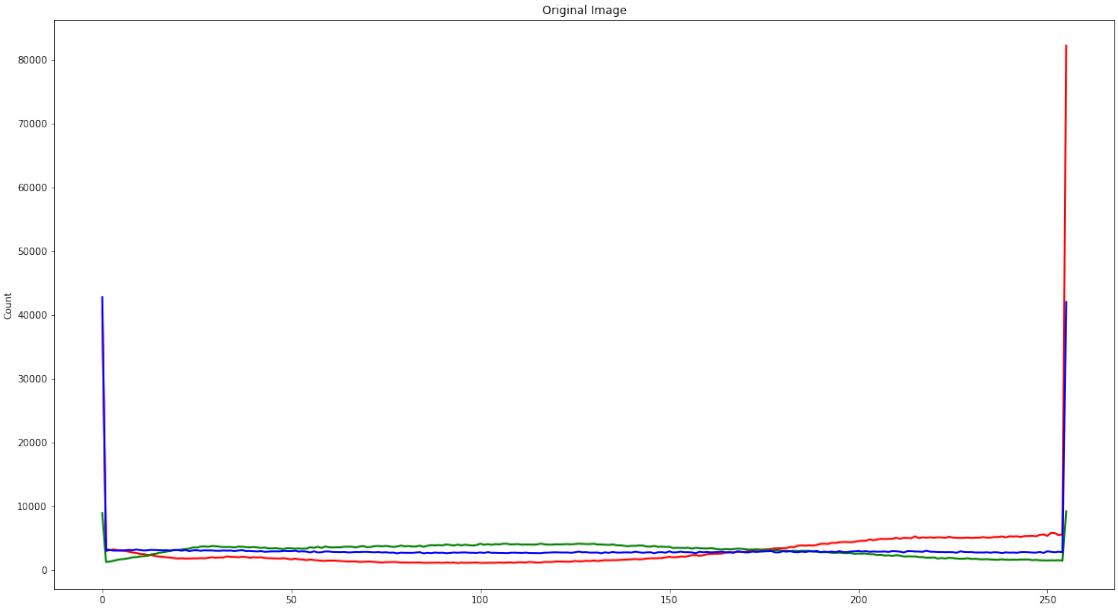
```
F:\apps\Anaconda\lib\site-packages\ipykernel_launcher.py:5:
MatplotlibDeprecationWarning: Adding an axes using the same arguments as a
previous axes currently reuses the earlier instance.  In a future version, a new
instance will always be created and returned.  Meanwhile, this warning can be
suppressed, and the future behavior ensured, by passing a unique label to each
axes instance.
  """
```

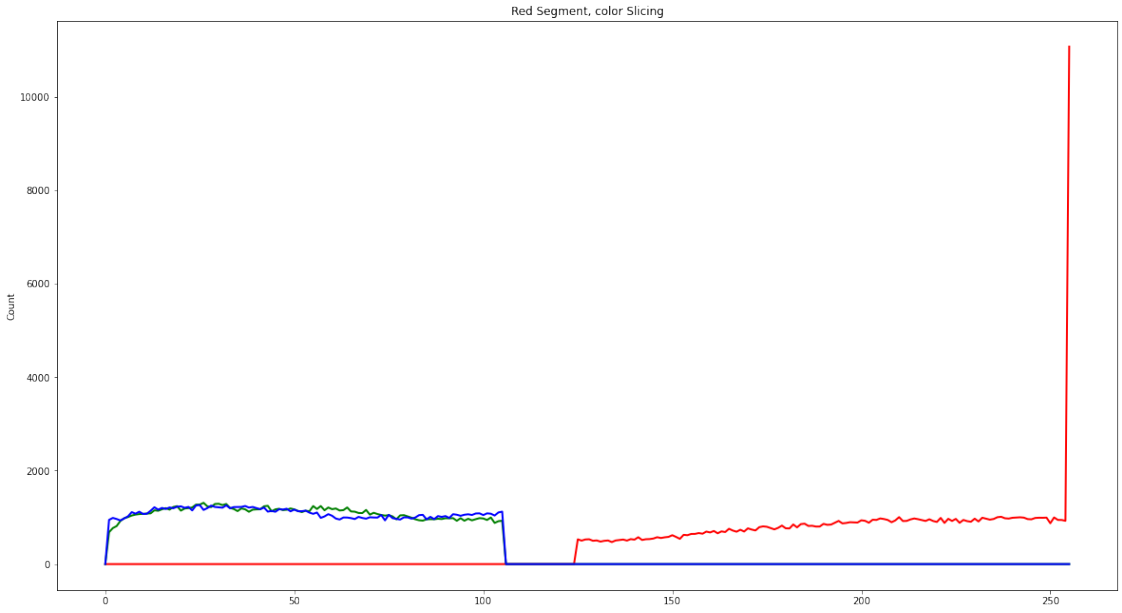
[20]:
```
(<AxesSubplot:title={'center':'Segmentation of red component using color
slicing'}>,
 <matplotlib.image.AxesImage at 0x25dab410e88>,
 Text(0.5, 1.0, 'Segmentation of red component using color slicing'),
 (-0.5, 899.5, 899.5, -0.5))
```



2.3.1 Comparing Histograms:

[21]:
```
rgb_histogram(rgb, "Original Image"), rgb_histogram(cs, "Red Segment, color␣
↪Slicing", True)
```





[21]: (None, None)

3 Q3

[22]:
```
rgb = cv2.imread('test1.png', cv2.IMREAD_COLOR)[:,:,::-1]
hsi = rgb_to_hsi(rgb)
print(rgb.shape)
plt.figure(figsize=(20, 20))
plt.imshow(rgb)
plt.show()
```

```
(659, 714, 3)
```

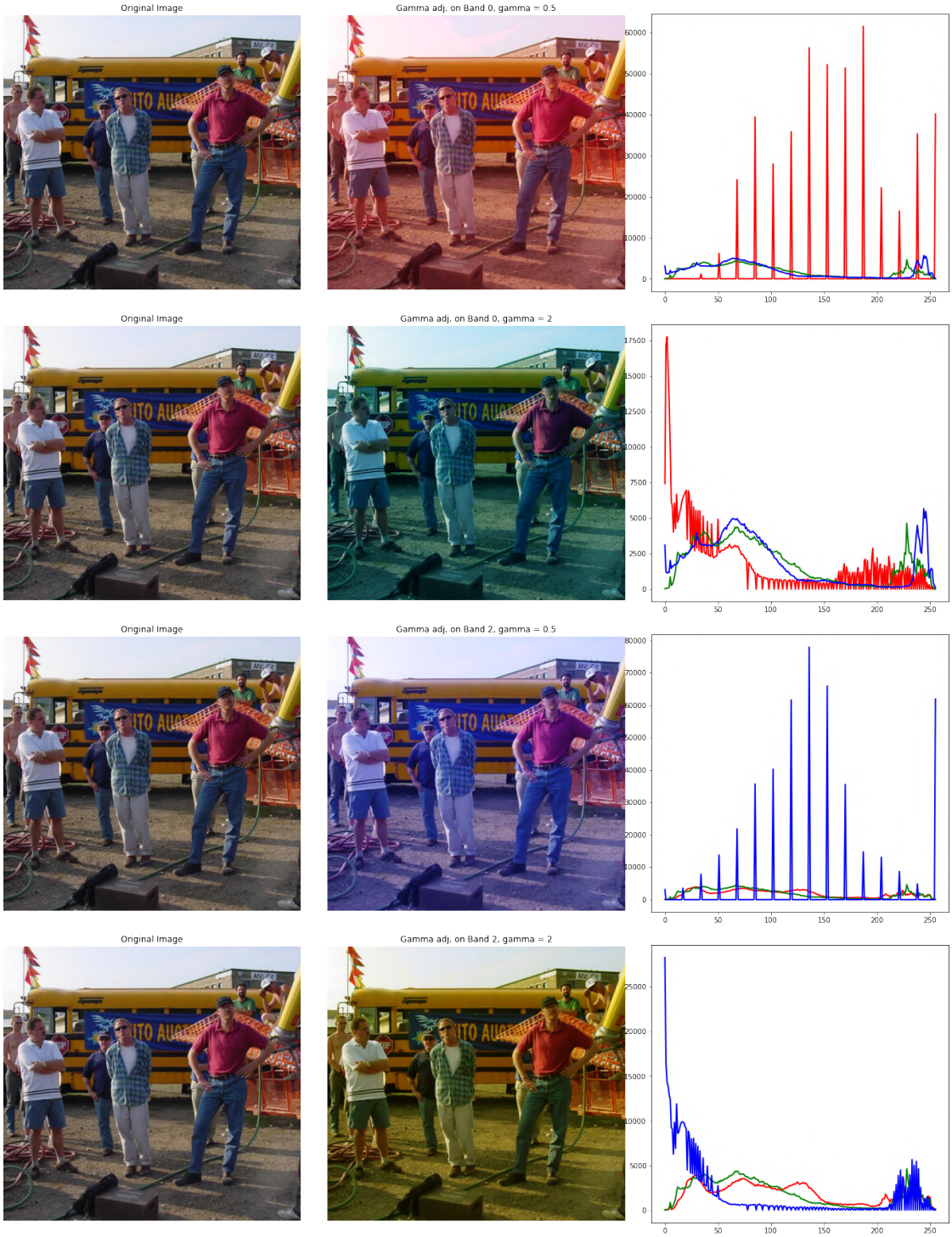
3.1 Gamma Adjustment

[23]:
```
def gamma_adjustment(rgb, gamma, channel):
    result = np.copy(rgb).astype(np.uint64)
    result[:, :, channel] = result[:, :, channel] ** gamma
    min_channel = np.amin(result[:, :, channel])
    max_channel = np.amax(result[:, :, channel])
    result[:, :, channel] = (result[:, :, channel] - min_channel) /␣
↪(max_channel - min_channel) * 255
    return result
```

[24]:
```
channels = [0, 2]
gammas = [0.5, 2]
clw = 2
```

```
plt.figure(figsize=(20, 25))
plt.subplot(4, 3, 1)
plt.tight_layout(pad=3.00)
index = 0
for channel in channels:
    for gamma in gammas:
        index+=1
        plt.subplot(4, 3, index),plt.imshow(rgb), plt.title('Original Image'),␣
 ↪plt.axis('off')

        index+=1

        g_adj = gamma_adjustment(rgb, gamma, channel)
        plt.subplot(4, 3, index),plt.imshow(g_adj), plt.title('Gamma adj, on␣
 ↪Band {}, gamma = {}'.format(channel, gamma)), plt.axis('off')
        index+=1
        r_map, g_map, b_map = rgb_map(g_adj)
        plt.subplot(4, 3, index),plt.plot(r_map, color='r', label = 'Red',␣
 ↪linewidth=clw), plt.plot(g_map, color='g', label = 'Green', linewidth=clw),␣
 ↪plt.plot(b_map, color='b', label = 'Blue', linewidth=clw)


plt.show()
```

```
F:\apps\Anaconda\lib\site-packages\ipykernel_launcher.py:12:
MatplotlibDeprecationWarning: Adding an axes using the same arguments as a
previous axes currently reuses the earlier instance.  In a future version, a new
instance will always be created and returned.  Meanwhile, this warning can be
suppressed, and the future behavior ensured, by passing a unique label to each
axes instance.
  if sys.path[0] == '':
```


Original Image
Gamma adj. on Band 0, gamma = 0.5
Original Image
Gamma adj. on Band 0, gamma = 2
Original Image
Gamma adj. on Band 2, gamma = 0.5
Original Image
Gamma adj. on Band 2, gamma = 2

3.2 Saturation Adjustment

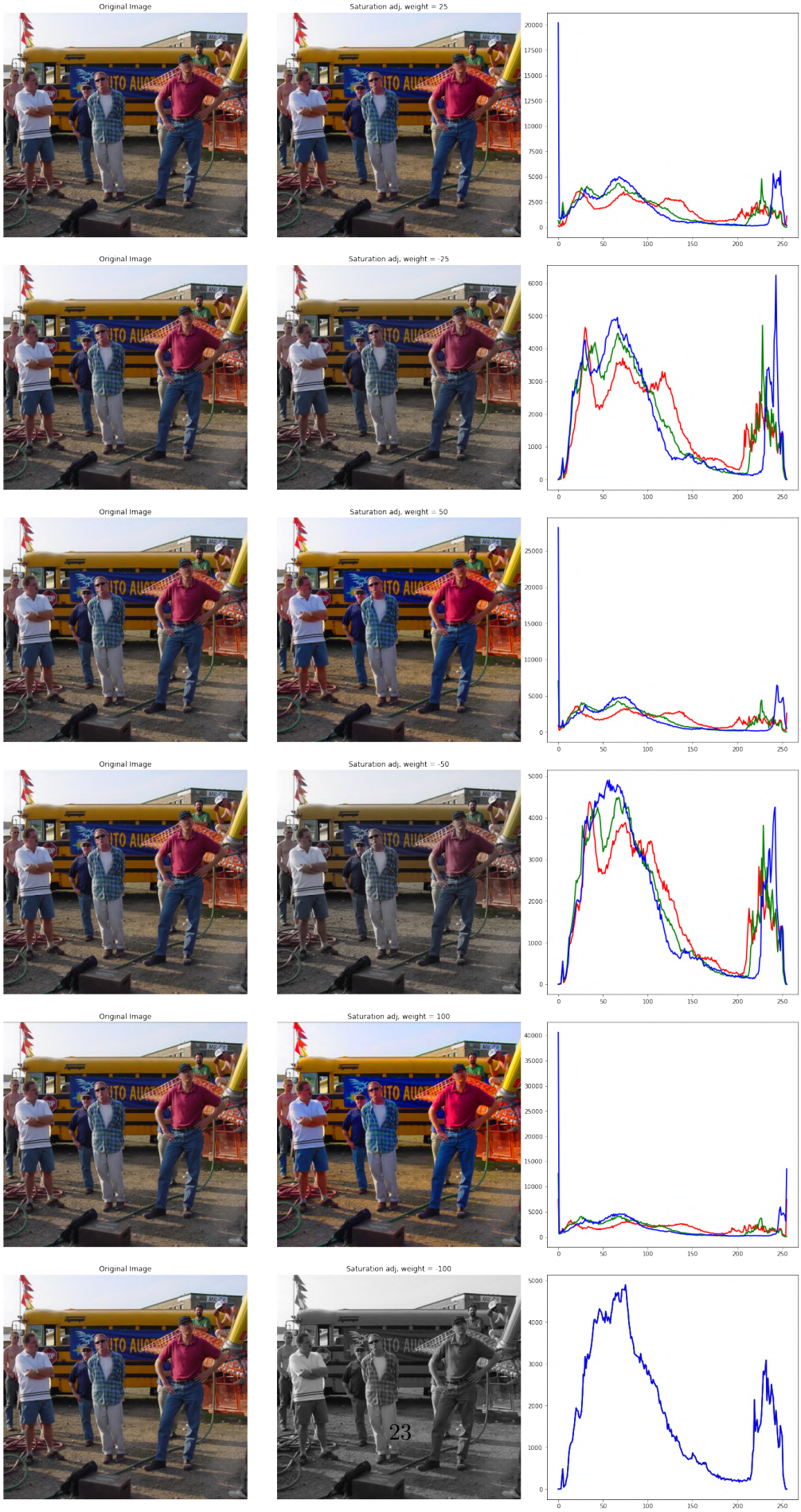
[25]:
```
def saturation_adjustment(hsi, sat_adj):
    res = hsi.copy()
    res[:, :, 1] = res[:, :, 1] * (1 + sat_adj/100)
    np.clip(res[:, :, 1], 0, 1, out=res[:, :, 1])
    print(np.amax(res[:, :, 1]))
    return res
```

[26]:
```
sat_weights = [25, -25, 50, -50, 100, -100]
clw = 2

plt.figure(figsize=(20, 36))
plt.subplot(6, 3, 1)
plt.tight_layout(pad=3.00)

index = 0
for sat_w in sat_weights:
    index+=1
    plt.subplot(6, 3, index), plt.imshow(rgb), plt.title('Original Image'), plt.
 ↪axis('off')
    index+=1

    sat_adj_res = saturation_adjustment(hsi, sat_w)

    rgb_sat_adj = hsi_to_rgb(sat_adj_res)
    plt.subplot(6, 3, index),plt.imshow(rgb_sat_adj), plt.title('Saturation␣
 ↪adj, weight = {}'.format(sat_w)), plt.axis('off')
    index+=1
    r_map, g_map, b_map = rgb_map(rgb_sat_adj)
    plt.subplot(6, 3, index),plt.plot(r_map, color='r', label = 'Red',␣
 ↪linewidth=clw), plt.plot(g_map, color='g', label = 'Green', linewidth=clw),␣
 ↪plt.plot(b_map, color='b', label = 'Blue', linewidth=clw)


plt.show()
```

```
F:\apps\Anaconda\lib\site-packages\ipykernel_launcher.py:11:
MatplotlibDeprecationWarning: Adding an axes using the same arguments as a
previous axes currently reuses the earlier instance.  In a future version, a new
instance will always be created and returned.  Meanwhile, this warning can be
suppressed, and the future behavior ensured, by passing a unique label to each
axes instance.
  # This is added back by InteractiveShellApp.init_path()
```

```
1.0
283.9999999999999 259.0833333333333 257.08333333333337
0.75
```

```
257.1599246526724 253.83333333333337 254.66666666666669
1.0
315.9999999999999 263.1666666666667 261.33333333333326
0.5
254.95465662035633 253.66666666666669 254.33333333333334
1.0
354.58333333333326 271.3333333333333 271.66666666666663
0.0
254.0 254.0 254.0
```


Original Image
Saturation adj. weight = 25
Original Image
Saturation adj. weight = -25
Original Image
Saturation adj. weight = 50
Original Image
Saturation adj. weight = -50
Original Image
Saturation adj. weight = 100
Original Image
Saturation adj. weight = -100

3.3 Hue Shifting

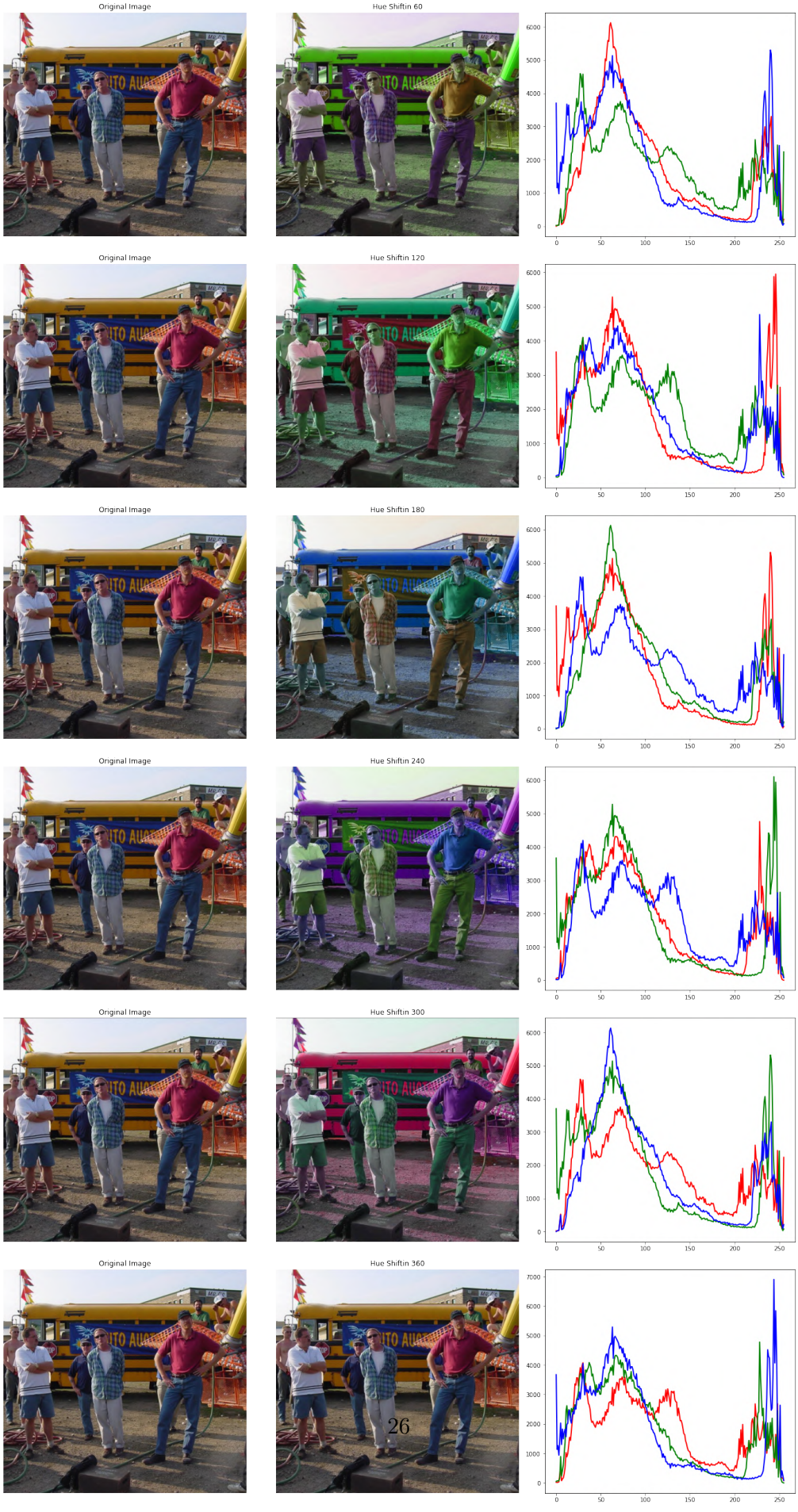
[27]:
```
def hue_shifting(hsi, degree):
    res = hsi.copy()
    res[:, :, 0] = (res[:, :, 0] + degree) % 360
    print(np.amax(res[:, :, 0]), np.amin(res[:, :, 0]))
    return res
```

[28]:
```
hue_shifts = [60, 120, 180, 240, 300, 360]
clw = 2

plt.figure(figsize=(20, 36))
plt.subplot(6, 3, 1)
plt.tight_layout(pad=3.00)

index = 0
for hue_s in hue_shifts:
    index+=1
    plt.subplot(6, 3, index), plt.imshow(rgb), plt.title('Original Image'), plt.
 ↪axis('off')
    index+=1

    hue_shift_res = hue_shifting(hsi, hue_s)

    rgb_hue_shift = hsi_to_rgb(hue_shift_res)
    plt.subplot(6, 3, index),plt.imshow(rgb_hue_shift), plt.title('Hue Shiftin␣
 ↪{}'.format(hue_s)), plt.axis('off')
    index+=1
    r_map, g_map, b_map = rgb_map(rgb_hue_shift)
    plt.subplot(6, 3, index),plt.plot(r_map, color='r', label = 'Red',␣
 ↪linewidth=clw), plt.plot(g_map, color='g', label = 'Green', linewidth=clw),␣
 ↪plt.plot(b_map, color='b', label = 'Blue', linewidth=clw)


plt.show()
```

```
F:\apps\Anaconda\lib\site-packages\ipykernel_launcher.py:11:
MatplotlibDeprecationWarning: Adding an axes using the same arguments as a
previous axes currently reuses the earlier instance.  In a future version, a new
instance will always be created and returned.  Meanwhile, this warning can be
suppressed, and the future behavior ensured, by passing a unique label to each
axes instance.
  # This is added back by InteractiveShellApp.init_path()
```

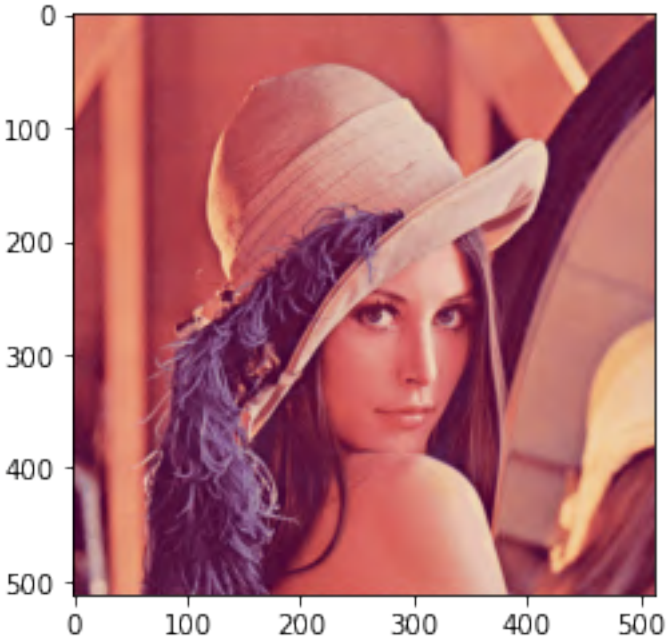
```
358.95549255436686 0.0
```

```
272.0 324.8947368421053 266.08000000000004
359.3911934584115 0.0
255.00000000000003 259.5465662035632 254.99999999999997
358.97702416567176 0.0
266.08000000000004 272.0 324.8947368421053
358.88509353480976 0.0
254.99999999999997 255.00000000000003 259.5465662035632
359.07261182614707 0.0
324.8947368421053 266.08000000000004 272.0
359.60463097203035 0.0
259.5465662035632 254.99999999999997 255.00000000000003
```


Original Image
Hue Shiftin 60
Original Image
Hue Shiftin 120
Original Image
Hue Shiftin 180
Original Image
Hue Shiftin 240
Original Image
Hue Shiftin 300
Original Image
Hue Shiftin 360

[29]:
```
rgb = cv2.imread('lena.bmp', cv2.IMREAD_COLOR)[:,:,::-1]
hsi = rgb_to_hsi(rgb)
print(rgb.shape)
plt.imshow(rgb)
plt.show()
```

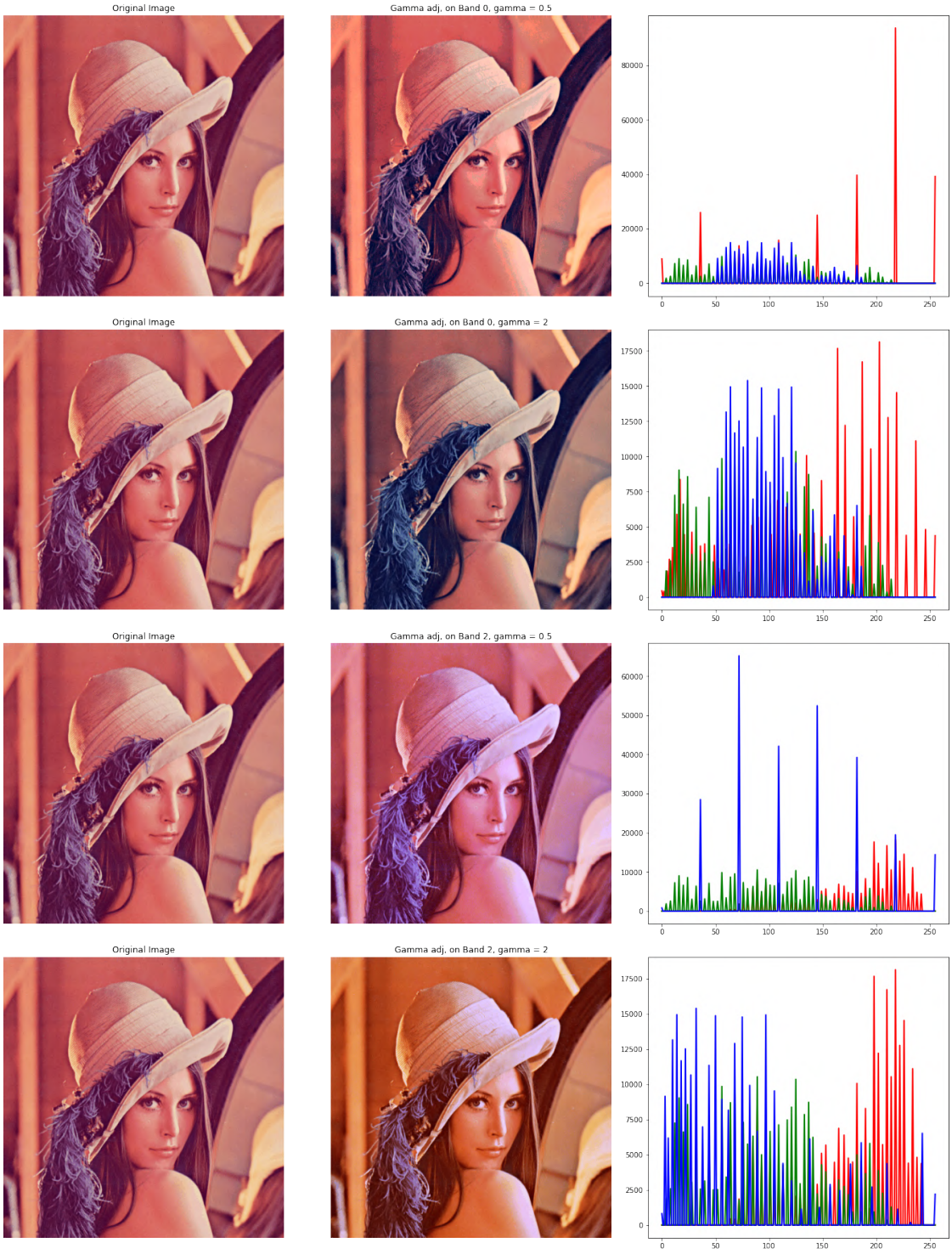
```
(512, 512, 3)
```



[30]:
```
channels = [0, 2]
gammas = [0.5, 2]
clw = 2

plt.figure(figsize=(20, 25))
plt.subplot(4, 3, 1)
plt.tight_layout(pad=3.00)
index = 0
for channel in channels:
    for gamma in gammas:
        index+=1
        plt.subplot(4, 3, index),plt.imshow(rgb), plt.title('Original Image'),␣
 ↪plt.axis('off')

        index+=1
```

```
        g_adj = gamma_adjustment(rgb, gamma, channel)
        plt.subplot(4, 3, index),plt.imshow(g_adj), plt.title('Gamma adj, on␣
↪Band {}, gamma = {}'.format(channel, gamma)), plt.axis('off')
        index+=1
        r_map, g_map, b_map = rgb_map(g_adj)
        plt.subplot(4, 3, index),plt.plot(r_map, color='r', label = 'Red',␣
↪linewidth=clw), plt.plot(g_map, color='g', label = 'Green', linewidth=clw),␣
↪plt.plot(b_map, color='b', label = 'Blue', linewidth=clw)


plt.show()
```

```
F:\apps\Anaconda\lib\site-packages\ipykernel_launcher.py:12:
MatplotlibDeprecationWarning: Adding an axes using the same arguments as a
previous axes currently reuses the earlier instance.  In a future version, a new
instance will always be created and returned.  Meanwhile, this warning can be
suppressed, and the future behavior ensured, by passing a unique label to each
axes instance.
  if sys.path[0] == '':
```

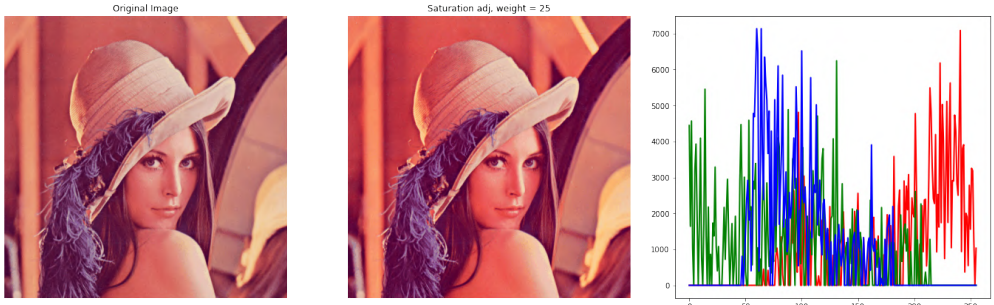


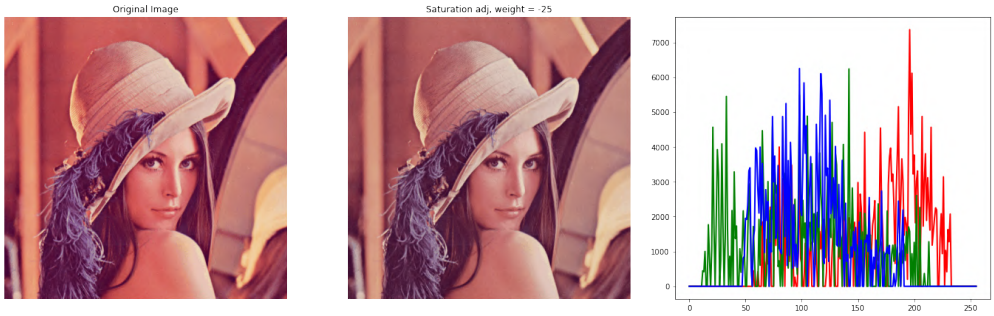
[31]:
```
sat_weights = [25, -25, 50, -50, 100, -100]
clw = 2
```

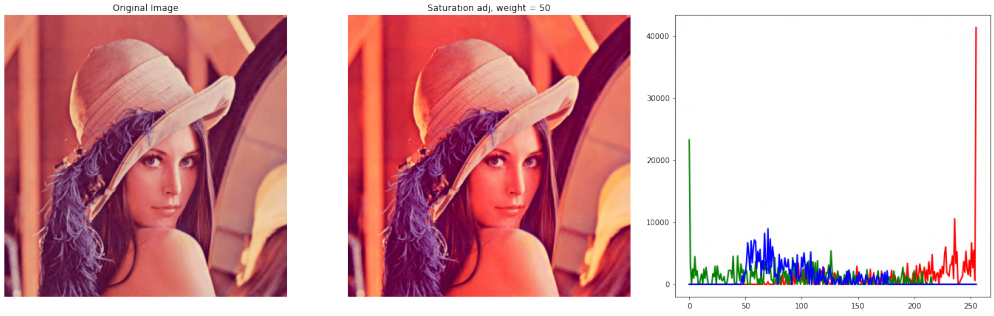
```
plt.figure(figsize=(20, 36))
plt.subplot(6, 3, 1)
plt.tight_layout(pad=3.00)

index = 0
for sat_w in sat_weights:
    index+=1
    plt.subplot(6, 3, index), plt.imshow(rgb), plt.title('Original Image'), plt.
 ↪axis('off')
    index+=1

    sat_adj_res = saturation_adjustment(hsi, sat_w)

    rgb_sat_adj = hsi_to_rgb(sat_adj_res)
    plt.subplot(6, 3, index),plt.imshow(rgb_sat_adj), plt.title('Saturation␣
 ↪adj, weight = {}'.format(sat_w)), plt.axis('off')
    index+=1
    r_map, g_map, b_map = rgb_map(rgb_sat_adj)
    plt.subplot(6, 3, index),plt.plot(r_map, color='r', label = 'Red',␣
 ↪linewidth=clw), plt.plot(g_map, color='g', label = 'Green', linewidth=clw),␣
 ↪plt.plot(b_map, color='b', label = 'Blue', linewidth=clw)


plt.show()
```

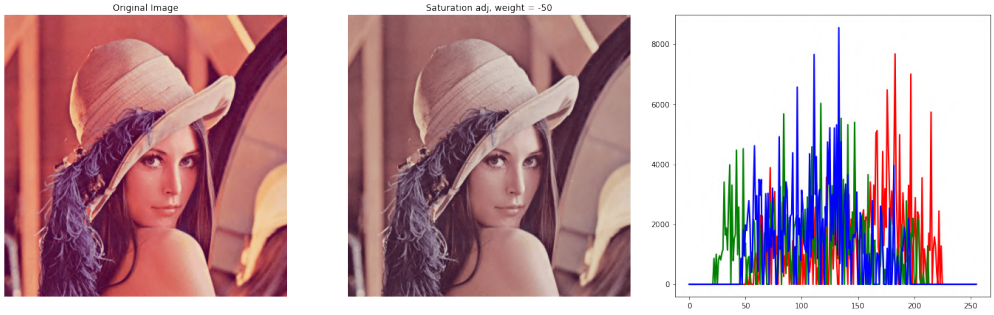
```
F:\apps\Anaconda\lib\site-packages\ipykernel_launcher.py:11:
MatplotlibDeprecationWarning: Adding an axes using the same arguments as a
previous axes currently reuses the earlier instance.  In a future version, a new
instance will always be created and returned.  Meanwhile, this warning can be
suppressed, and the future behavior ensured, by passing a unique label to each
axes instance.
  # This is added back by InteractiveShellApp.init_path()
```

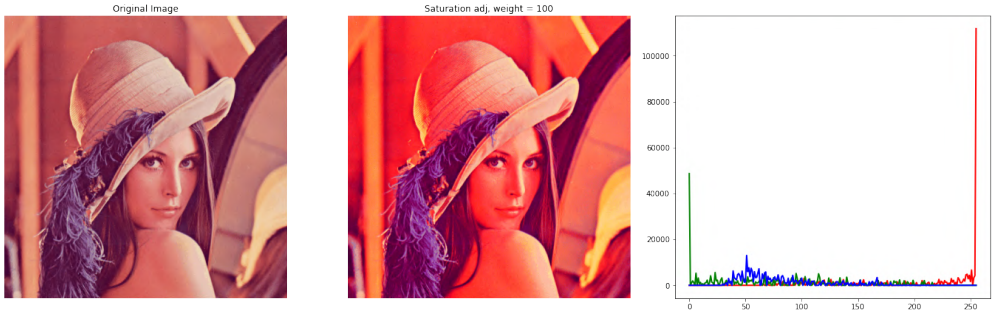
```
1.0
255.41666666666666 214.6666666666667 181.33333333333331
0.6838235294117647
232.25 213.3333333333333 190.66666666666666
1.0
271.66666666666663 215.3333333333333 176.66666666666666
0.45588235294117646
224.66666666666669 212.66666666666663 196.66666666666669
1.0
309.33333333333326 216.66666666666666 179.66666666666666
0.0
211.33333333333334 211.3333333333333 211.33333333333334
```


Original Image
Saturation adj, weight = 25


Original Image
Saturation adj, weight = -25


Original Image
Saturation adj, weight = 50


Original Image
Saturation adj, weight = -50


Original Image
Saturation adj, weight = 100

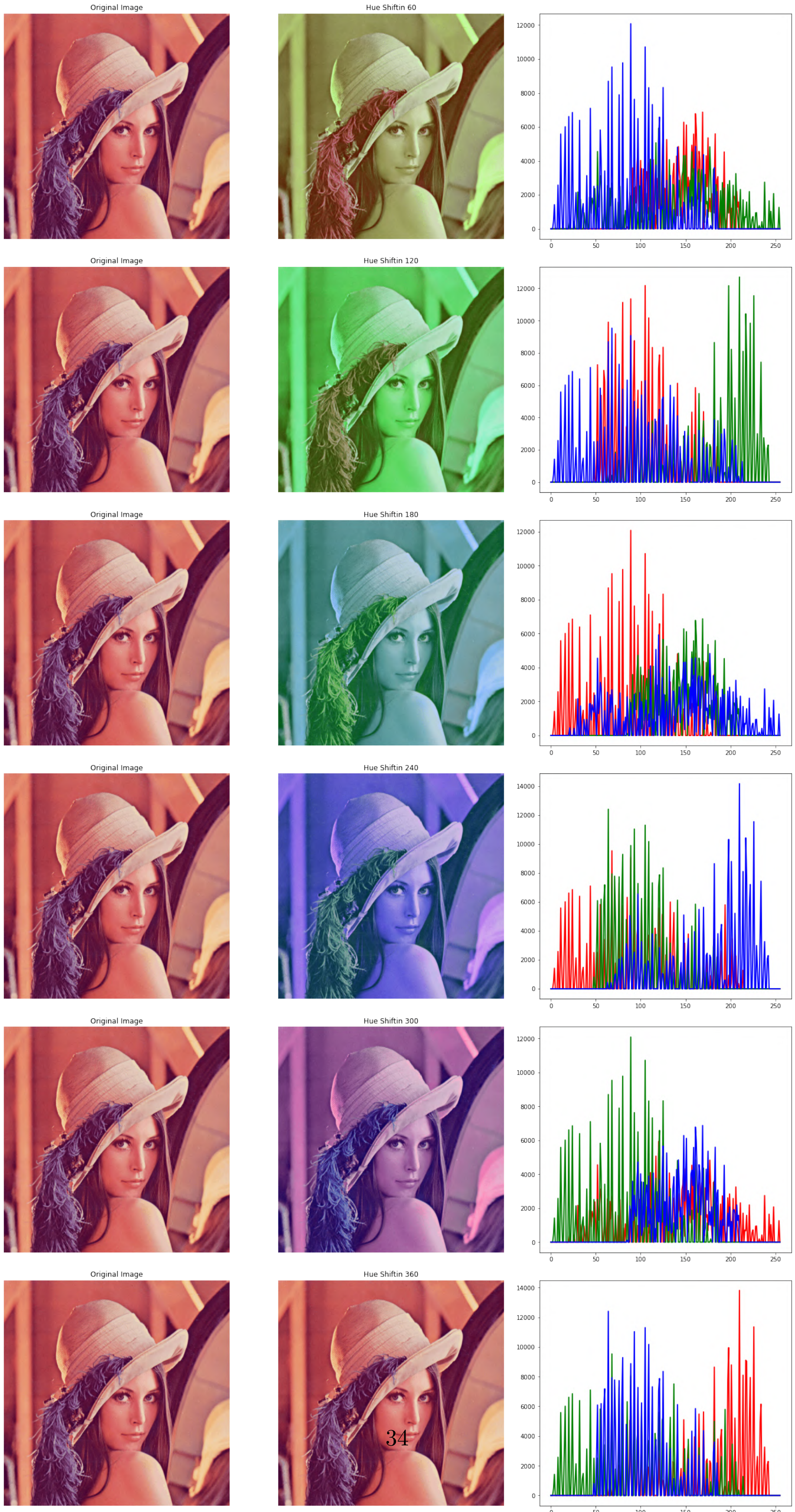

Original Image
Saturation adj, weight = -100

[32]:
```
hue_shifts = [60, 120, 180, 240, 300, 360]
clw = 2

plt.figure(figsize=(20, 36))
plt.subplot(6, 3, 1)
plt.tight_layout(pad=3.00)

index = 0
for hue_s in hue_shifts:
    index+=1
    plt.subplot(6, 3, index), plt.imshow(rgb), plt.title('Original Image'), plt.
 ↪axis('off')
    index+=1

    hue_shift_res = hue_shifting(hsi, hue_s)

    rgb_hue_shift = hsi_to_rgb(hue_shift_res)
    plt.subplot(6, 3, index),plt.imshow(rgb_hue_shift), plt.title('Hue Shiftin␣
 ↪{}'.format(hue_s)), plt.axis('off')
    index+=1
    r_map, g_map, b_map = rgb_map(rgb_hue_shift)
    plt.subplot(6, 3, index),plt.plot(r_map, color='r', label = 'Red',␣
 ↪linewidth=clw), plt.plot(g_map, color='g', label = 'Green', linewidth=clw),␣
 ↪plt.plot(b_map, color='b', label = 'Blue', linewidth=clw)


plt.show()
```

```
F:\apps\Anaconda\lib\site-packages\ipykernel_launcher.py:11:
MatplotlibDeprecationWarning: Adding an axes using the same arguments as a
previous axes currently reuses the earlier instance.  In a future version, a new
instance will always be created and returned.  Meanwhile, this warning can be
suppressed, and the future behavior ensured, by passing a unique label to each
axes instance.
  # This is added back by InteractiveShellApp.init_path()
```

```
354.79128089714493 3.8857649122587645
209.82608695652175 254.03649635036493 185.99999999999997
155.7778140857061 48.2584690041092
185.99999999999997 242.00000000000003 213.99999999999997
215.7778140857061 108.2584690041092
185.99999999999997 209.82608695652175 254.03649635036493
275.7778140857061 168.2584690041092
214.0 185.99999999999997 242.00000000000006
335.7778140857061 228.2584690041092
```

```
254.03649635036493 185.99999999999997 209.82608695652175
358.3867808859061 0.0
242.00000000000006 214.0 185.99999999999997
```


Original Image
Hue Shiftin 60
Original Image
Hue Shiftin 120
Original Image
Hue Shiftin 180
Original Image
Hue Shiftin 240
Original Image
Hue Shiftin 300
Original Image
Hue Shiftin 360

[ ]:

[ ]: