Abby Ortego
CMPS 470 – 01
Midterm Program – Kohonen SOFM

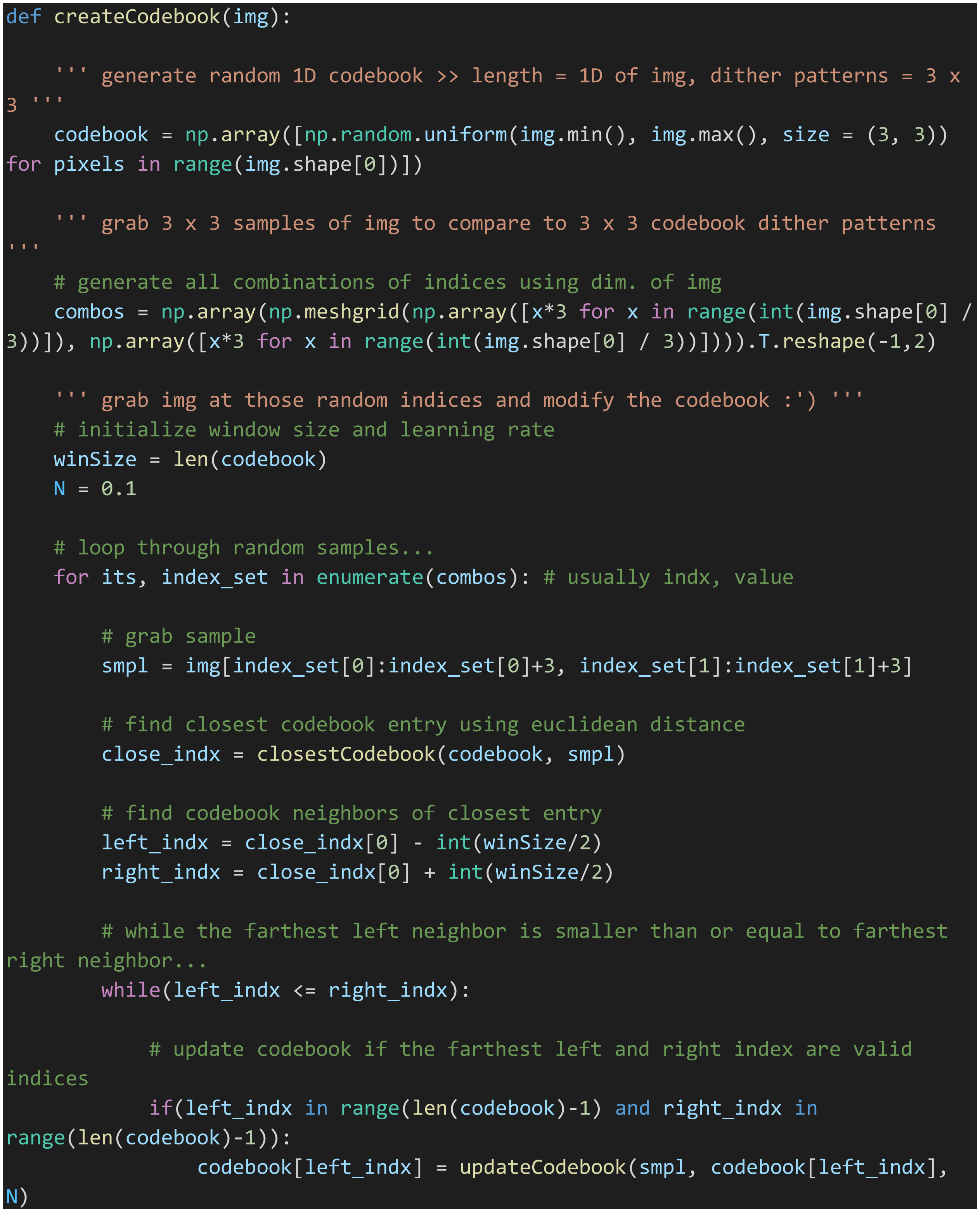
The following are imports:

```python
# %% IMPORTS
import numpy as np
from skimage import data, color
import matplotlib.pyplot as plt
import imageio as iio
import cv2
```

First, we call the ***compressImg*** function:

```python
def compressImg(img):

    ''' size img to be divisble by a 3x3 window '''
    while img.shape[0] % 3 != 0:
        img = np.delete(img, img.shape[0]-1, axis = 0)
        img = np.delete(img, img.shape[0]-1, axis = 1)
    #

    ''' create a space for the smol image '''
    smol_img = np.zeros([int(img.shape[0]/3), int(img.shape[0]/3)])

    ''' generate codebook for img '''
    codebook = createCodebook(img)

    ''' grab 3 x 3 regions of the img, find closest match, and compress '''
    for x,y in np.ndindex(smol_img.shape):
        region = img[x*3:x*3+3, y*3:y*3+3] # grabbing 3x3 samples of img
        closest_indx = closestCodebook(codebook, region)[0] # finding closest codebook entry

        # assign center of closest codebook entry to represent one pixel in smol img
        smol_img[x,y] = codebook[closest_indx][int(codebook[closest_indx].shape[0] / 2), int(codebook[closest_indx].shape[1] / 2)]
    #

    return smol_img
#
```

Which in turn calls the ***createCodebook*** function to train on the image:



```
def createCodebook(img):

    ''' generate random 1D codebook >> length = 1D of img, dither patterns = 3 x
3 '''
    codebook = np.array([np.random.uniform(img.min(), img.max(), size = (3, 3))
for pixels in range(img.shape[0])])

    ''' grab 3 x 3 samples of img to compare to 3 x 3 codebook dither patterns
'''
    # generate all combinations of indices using dim. of img
    combos = np.array(np.meshgrid(np.array([x*3 for x in range(int(img.shape[0] /
3))]), np.array([x*3 for x in range(int(img.shape[0] / 3))]))).T.reshape(-1,2)

    ''' grab img at those random indices and modify the codebook :') '''
    # initialize window size and learning rate
    winSize = len(codebook)
    N = 0.1

    # loop through random samples...
    for its, index_set in enumerate(combos): # usually indx, value

        # grab sample
        smpl = img[index_set[0]:index_set[0]+3, index_set[1]:index_set[1]+3]

        # find closest codebook entry using euclidean distance
        close_indx = closestCodebook(codebook, smpl)

        # find codebook neighbors of closest entry
        left_indx = close_indx[0] - int(winSize/2)
        right_indx = close_indx[0] + int(winSize/2)

        # while the farthest left neighbor is smaller than or equal to farthest
right neighbor...
        while(left_indx <= right_indx):

            # update codebook if the farthest left and right index are valid
indices
            if(left_indx in range(len(codebook)-1) and right_indx in
range(len(codebook)-1)):
                codebook[left_indx] = updateCodebook(smpl, codebook[left_indx],
N)
```

```
            #

            left_indx += 1
        #...fin

        # cut down window size and learning rate
        if(its % 8 == 0):
            if int(winSize/2) < 1:
                winSize = 1
            else:
                winSize = int(winSize/2)
            #
            N /= 2
        #

    #...fin

    return codebook
#
```

Within the ***createCodebook***, we apply the ***closestCodebook*** function to find the most similar codebook entry using Euclidean distance:

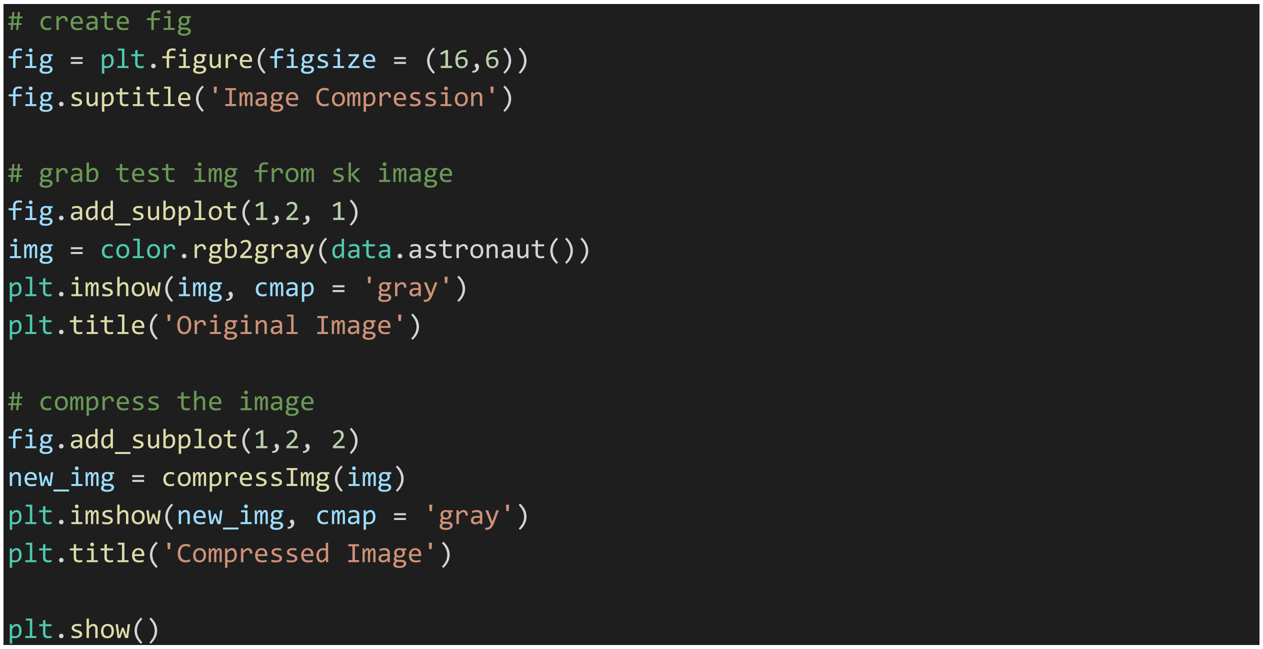
```
def closestCodebook(codebook, smpl):

    distance = np.array([np.linalg.norm(smpl - dither) for dither in codebook]) #
euclidean distance :D

    return np.where(distance == distance.min())[0] # return index of closest
codebook entry
#
```

The ***createCodebook*** also calls the ***updateCodebook*** which updates a certain section of the codebook by a particular learning rate (both determined by the number of iterations):

```
def updateCodebook(x, cluster, N):
    return cluster + (N * (x - cluster))
#
```

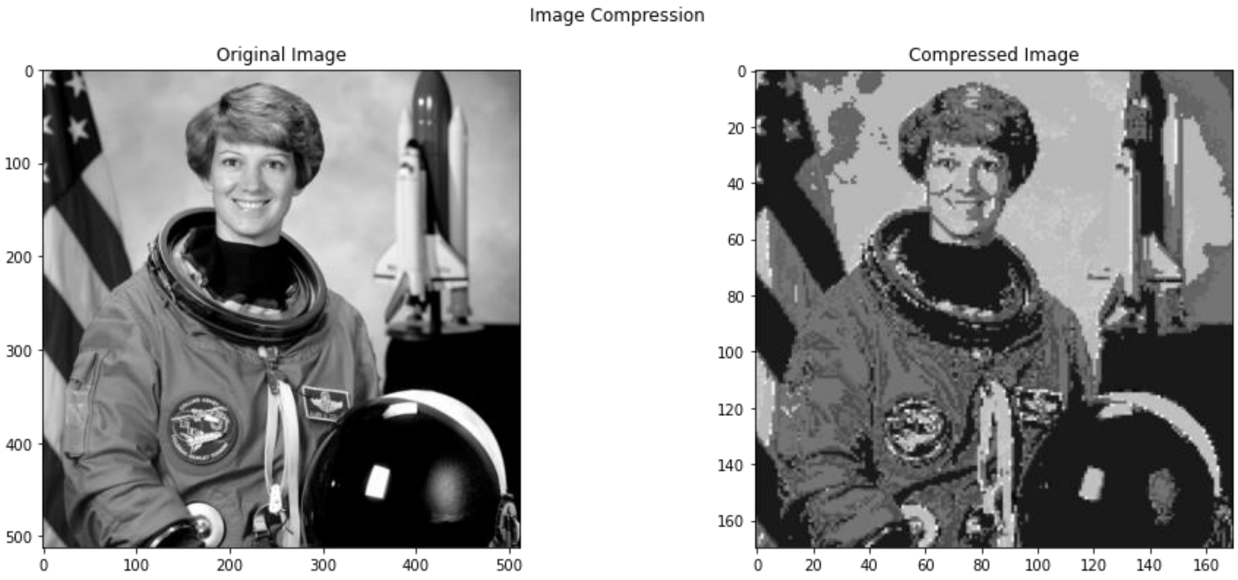
Abby Ortego
CMPS 470 – 01
Midterm Program – Kohonen SOFM

Compression of a black and white image…

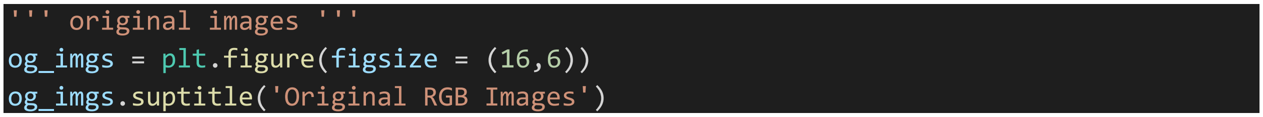


```
# create fig
fig = plt.figure(figsize = (16,6))
fig.suptitle('Image Compression')

# grab test img from sk image
fig.add_subplot(1,2, 1)
img = color.rgb2gray(data.astronaut())
plt.imshow(img, cmap = 'gray')
plt.title('Original Image')

# compress the image
fig.add_subplot(1,2, 2)
new_img = compressImg(img)
plt.imshow(new_img, cmap = 'gray')
plt.title('Compressed Image')

plt.show()
```

…with the following output:

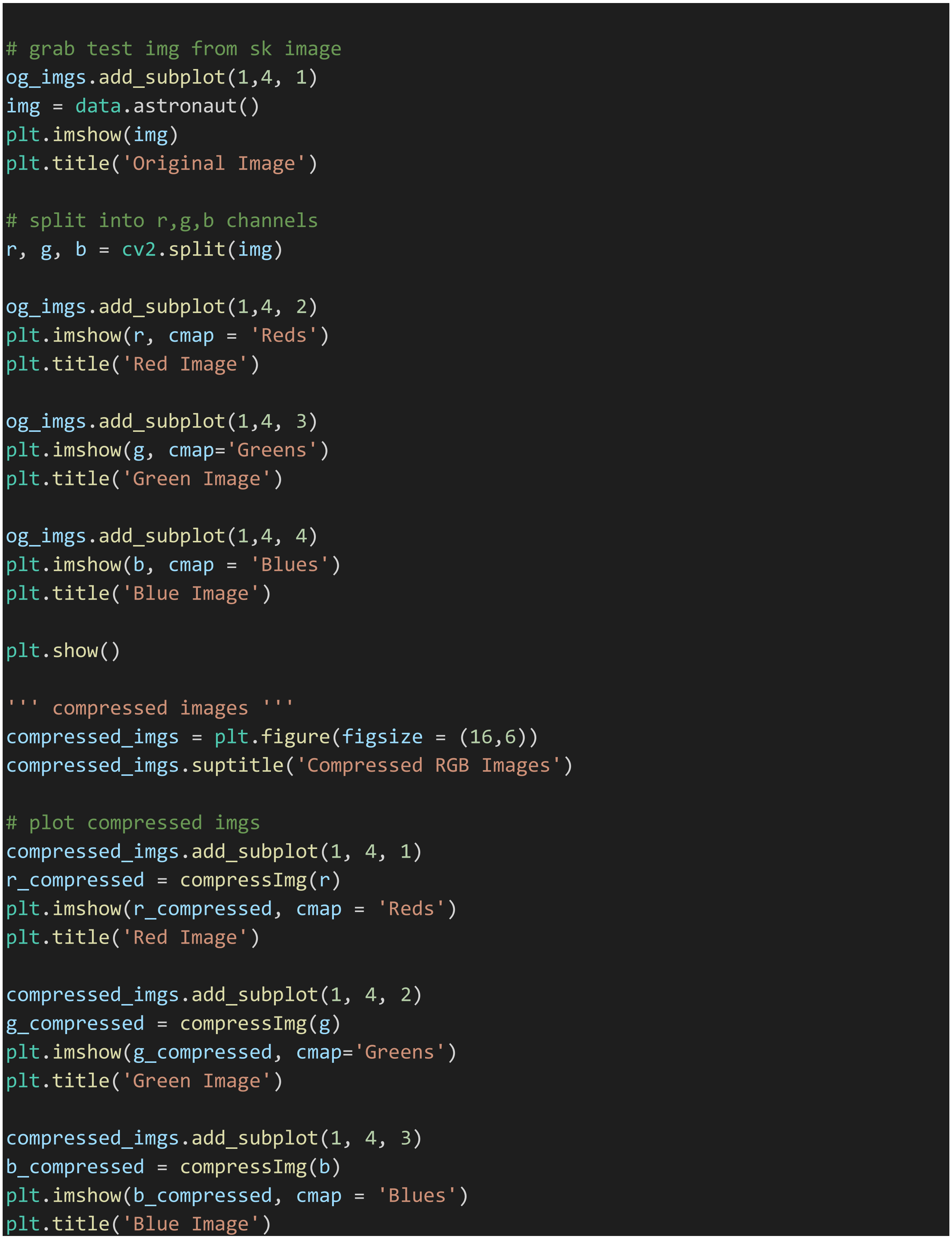


Compression of a color image…



```
''' original images '''
og_imgs = plt.figure(figsize = (16,6))
og_imgs.suptitle('Original RGB Images')
```

Abby Ortego
CMPS 470 – 01
Midterm Program – Kohonen SOFM

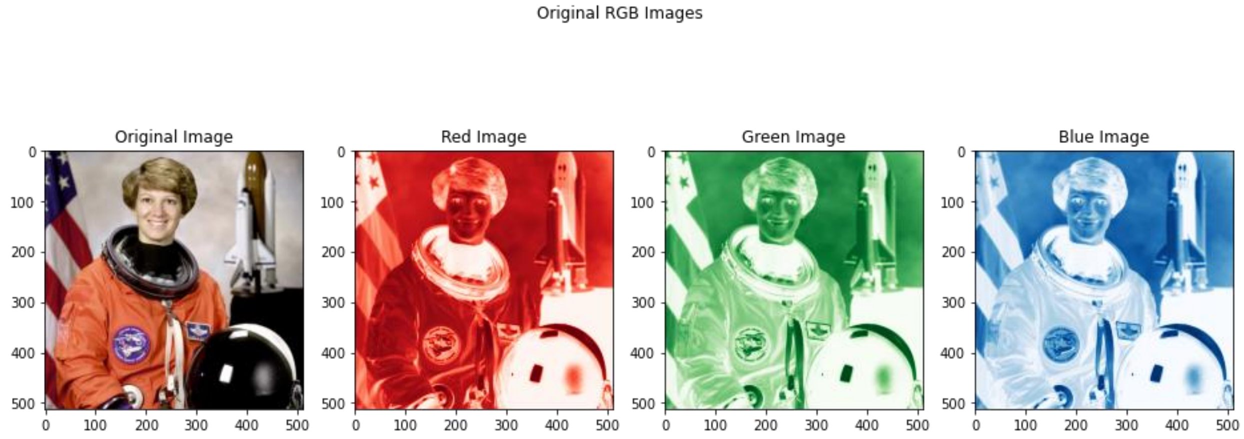


```
# grab test img from sk image
og_imgs.add_subplot(1,4, 1)
img = data.astronaut()
plt.imshow(img)
plt.title('Original Image')

# split into r,g,b channels
r, g, b = cv2.split(img)

og_imgs.add_subplot(1,4, 2)
plt.imshow(r, cmap = 'Reds')
plt.title('Red Image')

og_imgs.add_subplot(1,4, 3)
plt.imshow(g, cmap='Greens')
plt.title('Green Image')

og_imgs.add_subplot(1,4, 4)
plt.imshow(b, cmap = 'Blues')
plt.title('Blue Image')

plt.show()

''' compressed images '''
compressed_imgs = plt.figure(figsize = (16,6))
compressed_imgs.suptitle('Compressed RGB Images')

# plot compressed imgs
compressed_imgs.add_subplot(1, 4, 1)
r_compressed = compressImg(r)
plt.imshow(r_compressed, cmap = 'Reds')
plt.title('Red Image')

compressed_imgs.add_subplot(1, 4, 2)
g_compressed = compressImg(g)
plt.imshow(g_compressed, cmap='Greens')
plt.title('Green Image')

compressed_imgs.add_subplot(1, 4, 3)
b_compressed = compressImg(b)
plt.imshow(b_compressed, cmap = 'Blues')
plt.title('Blue Image')
```

Abby Ortego
CMPS 470 – 01
Midterm Program – Kohonen SOFM

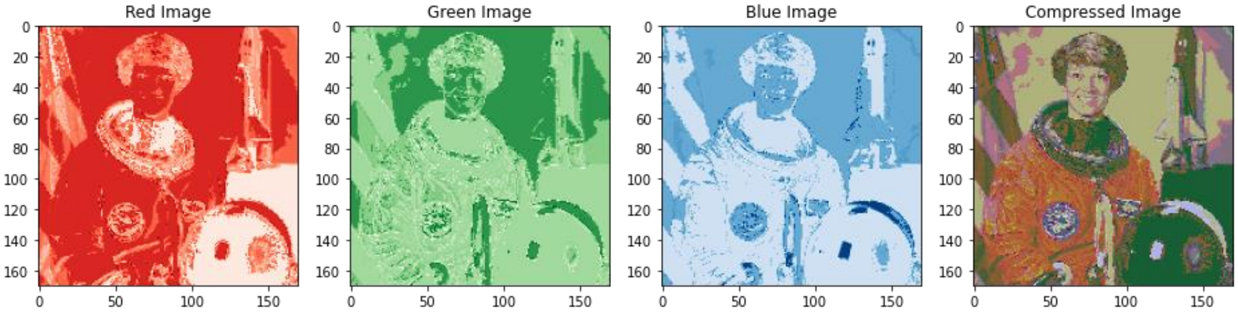
```
compressed_imgs.add_subplot(1, 4, 4)
combined_img = (cv2.merge([r_compressed, g_compressed, b_compressed])) / 255
plt.imshow(combined_img)
plt.title('Compressed Image')

plt.show()

''' side by side comparison '''
comparison = plt.figure(figsize = (16,6))
comparison.suptitle('Original v Compressed Color Image')

comparison.add_subplot(1, 2, 1)
plt.imshow(img)
plt.title('Original Image')

comparison.add_subplot(1, 2, 2)
plt.imshow(combined_img)
plt.title('Compressed Image')

plt.show()
```

…with the following output:



Abby Ortego
CMPS 470 – 01
Midterm Program – Kohonen SOFM

Compressed RGB Images

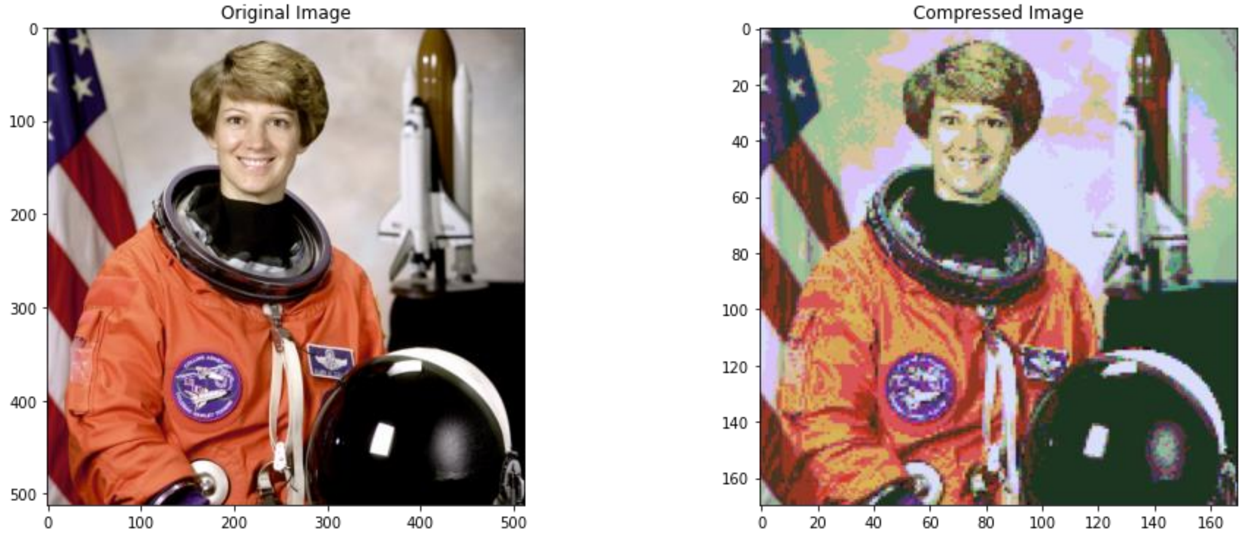

Red Image

Green Image

Blue Image

Compressed Image

Original v Compressed Color Image


Original Image

Compressed Image