hw0

January 30, 2018

1 Homework 0

In this homework, we will go through basic linear algebra and image manipulation using python to get everyone on the same page for the prerequisite skills for this class.

Hãy chy on code ngay bên di này trc import các modules cn thit !

```
In [1]: #Imports the print function from newer versions of python
        from __future__ import print_function

        #Setup

        # The Random module for implements pseudo-random number generators
        import random

        # Numpy is the main package for scientific computing with Python.
        # This will be one of our most used libraries in this class
        import numpy as np


        #Imports all the methods in each of the files: linalg.py and imageManip.py
        from linalg import *
        from imageManip import *


        #Matplotlib is a useful plotting library for python
        import matplotlib.pyplot as plt
        # This code is to make matplotlib figures appear inline in the
        # notebook rather than in a new window.
        %matplotlib inline
        plt.rcParams['figure.figsize'] = (10.0, 8.0) # set default size of plots
        plt.rcParams['image.interpolation'] = 'nearest'
        plt.rcParams['image.cmap'] = 'gray'

        # Some more magic so that the notebook will reload external python modules;
        # see http://stackoverflow.com/questions/1907993/autoreload-of-modules-in-ipython
        %load_ext autoreload
        %autoreload 2
```

```
%reload_ext autoreload

print("Huynh Bao Quoc (B1400516)") #ví d: Phm Nguyên Khang (1348)
print("Nguyen Trung Hau (B1400493)")
```

```
Huynh Bao Quoc (B1400516)
Nguyen Trung Hau (B1400493)
```

2 Question 1: Linear Algebra Review

Please implement all the required methods in linalg.py.

2.1 Question 1.1 (5 points)

Define the following using numpy:
(Hãy dùng numpy matrix i vi matrix và numpy array i vi vector. Xem thêm trong file linag.py)
M là ma trân 4 hàng 3 ct (4 x 3)
a là vector hàng (1 x 3)
b là vector ct (3 x 1)

$$M = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 10 & 11 & 12 \end{bmatrix}$$

$$a = \begin{bmatrix} 1 & 1 & 0 \end{bmatrix}$$

$$b = \begin{bmatrix} -1 \\ 2 \\ 5 \end{bmatrix}$$

```
In [4]: ### YOUR CODE HERE
        M=np.matrix([[1,2,3],[4,5,6],[7,8,9],[10,11,12]])
        a=np.array([[1,1,0]])
        b=np.array([[-1,2,5]]).T
        ### END CODE HERE
        print("M = \n", M)
        print("a = ", a)
        print("b = ", b)
```

```
M =
 [[ 1  2  3]
 [ 4  5  6]
 [ 7  8  9]
 [10 11 12]]
a =  [[1 1 0]]
b =  [[-1]
```

```
 [ 2]
 [ 5]]
```

2.2 Question 1.2 (5 points)

Implement the dot_product method in linalg.py and check that it returns the correct answer for $a^T b$.

```
In [4]: aDotB = dot_product(a, b)
        print (aDotB)
```

```
(1, 3)
(3, 1)
1
```

2.3 Question 1.3 (5 points)

Implement the matrix_mult method in linalg.py and use it to compute $(a^T b)Ma$ with a, b as column vectors, M is a matrix.

```
In [5]: ans = matrix_mult(M, a.T, b)
        print (ans)
```

```
[[ 3]
 [ 9]
 [15]
 [21]]
```

2.4 Question 4 (10 points)

Implement the get_singular_values method. In this method, perform singular value decomposition on the input matrix and return the largest n singular values (n specified in the method calls below).

```
In [6]: print(get_singular_values(M, 1))
        print(get_singular_values(M, 2))
```

```
None
None
```

2.5 Question 1.5 (10 points)

Implement the get_eigen_values_and_vectors method. In this method, perform eigen value decomposition on the following matrix and return the largest n eigen values and corresponding eigen vectors (n specified in the method calls below).

$$M = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

```
In [21]: M = np.matrix([[1,2,3],[4,5,6],[7,8,9]])
         val, vec = get_eigen_values_and_vectors(M[:,:3], 1)
         print("Values = \n", val)
         print("Vectors = \n", vec)
         val, vec = get_eigen_values_and_vectors(M[:,:3], 2)
         print("Values = \n", val)
         print("Vectors = \n", vec)
```

```
Values =
 [  1.61168440e+01  -1.11684397e+00  -9.75918483e-16]
Vectors =
 [[-0.23197069 -0.78583024  0.40824829]
 [-0.52532209 -0.08675134 -0.81649658]
 [-0.8186735   0.61232756  0.40824829]]
Values =
 [  1.61168440e+01  -1.11684397e+00  -9.75918483e-16]
Vectors =
 [[-0.23197069 -0.78583024  0.40824829]
 [-0.52532209 -0.08675134 -0.81649658]
 [-0.8186735   0.61232756  0.40824829]]
```

3 Part 2: Image Manipulation

```
In [6]: image1_path = './image1.jpg'
        image2_path = './image2.jpg'
```

3.1 Question 2.1 (5 points)

Implement the load method in imageManip.py and read the display method below. We will use these two methods through the rest of the notebook to visualize our work.

```
In [7]: def display(img):
            # Show image
            plt.imshow(img)
            plt.axis('off')
            plt.show()
```

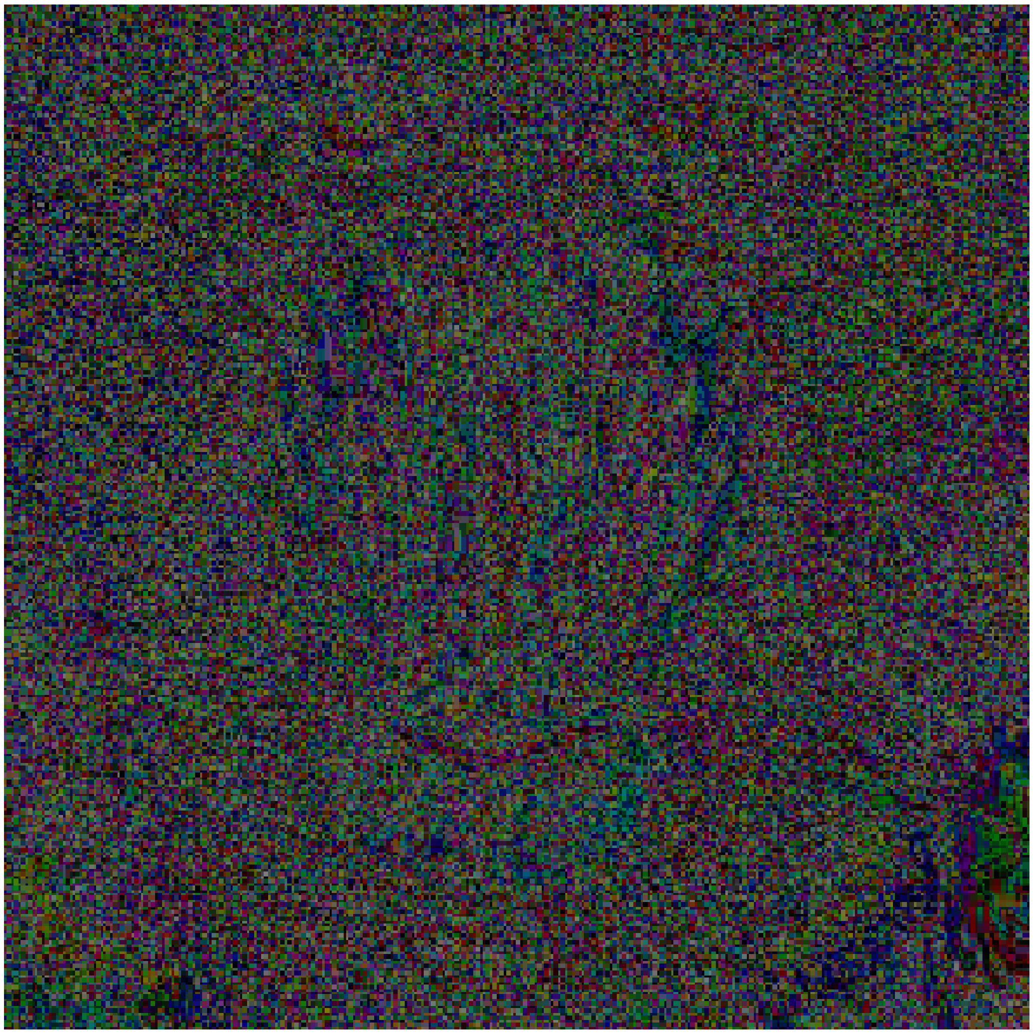
```
In [8]: image1 = load(image1_path)
        image2 = load(image2_path)

        display(image1)
        display(image2)
```

3.2 Question 2.2 (10 points)

Implement the change_value method by converting images according to $x_n = 0.5 * x_p^2$ for every pixel, where x_n is the new value and x_p is the original value.

```
In [9]: new_image = change_value(image1)
        display(new_image)
```

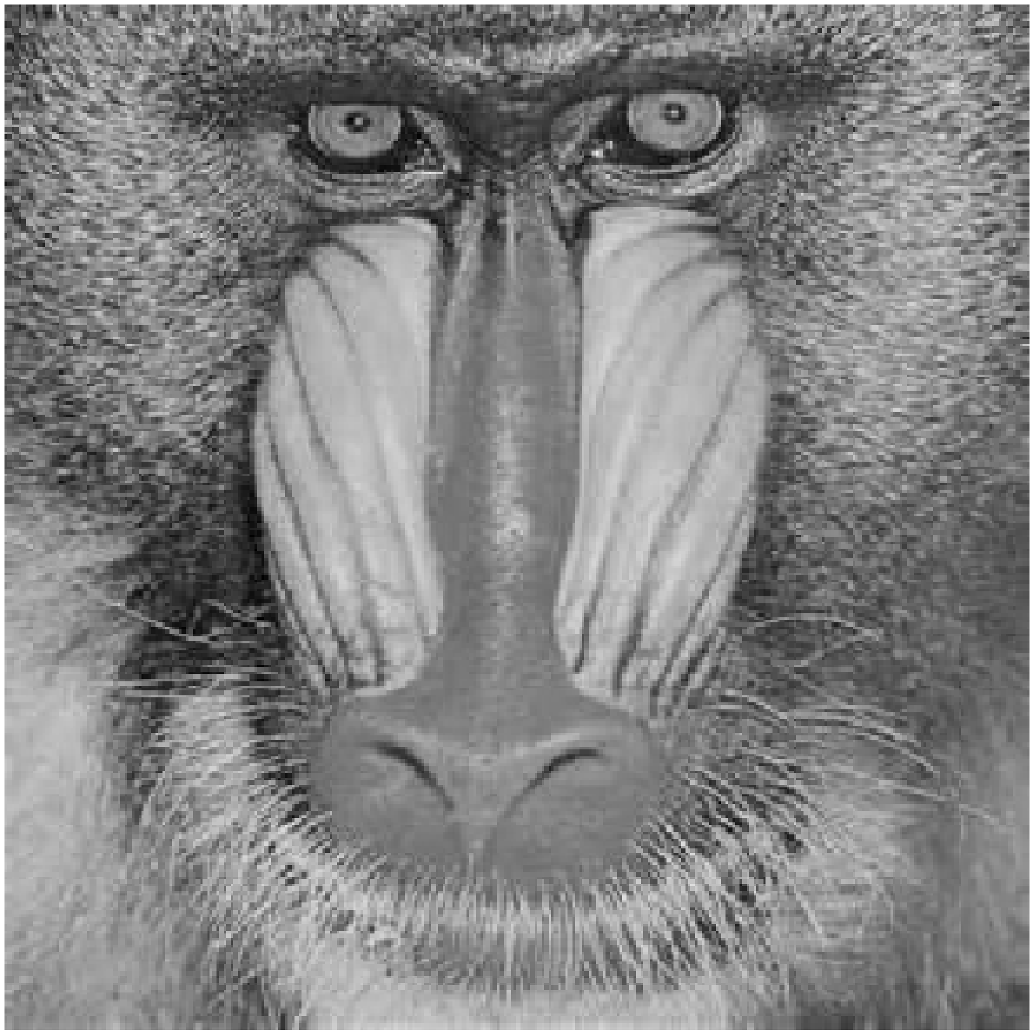


3.3 Question 2.3 (10 points)

Implement the convert_to_grey_scale method and convert the image into grey scale.

```
In [10]: image1 = load(image1_path)
         image2 = load(image2_path)
         grey_image = convert_to_grey_scale(image1)
         display(grey_image)
```

```
(300, 300, 3)
```

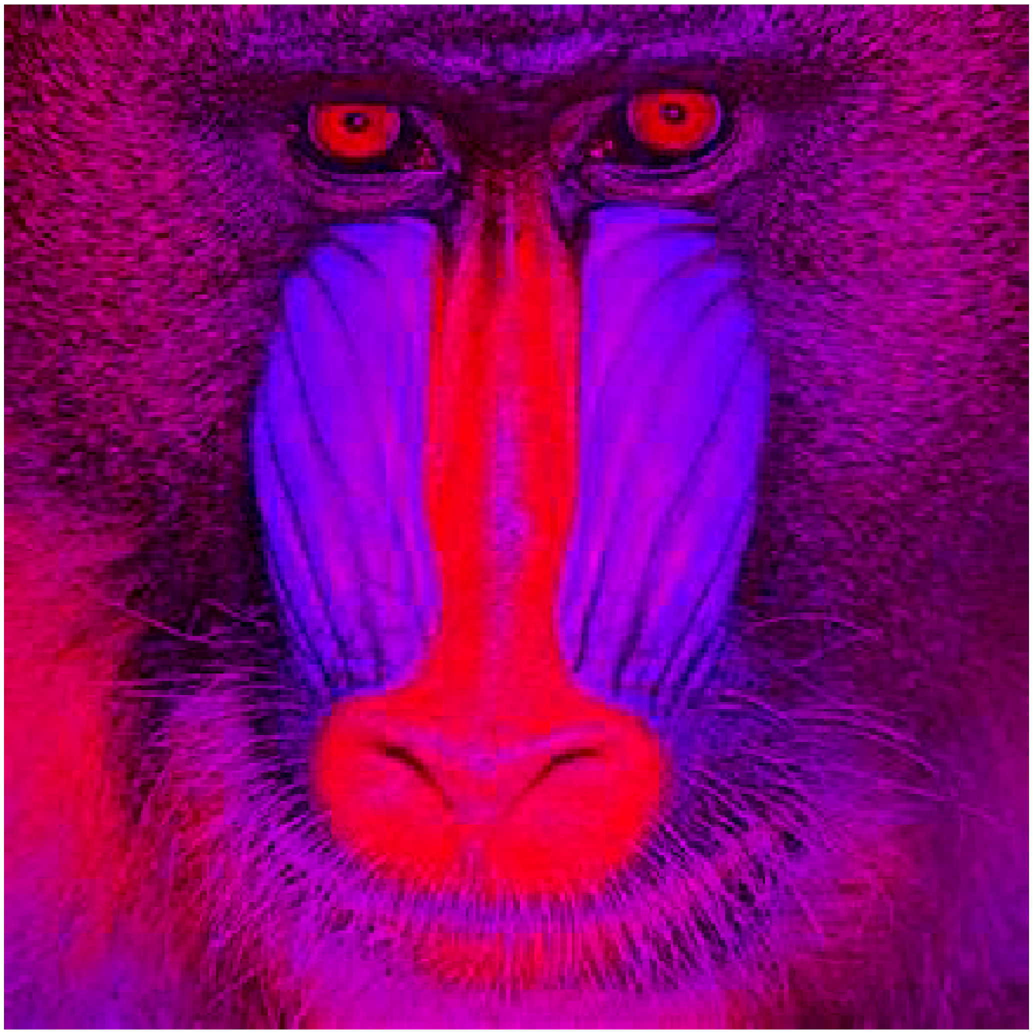


3.4 Question 2.4 (10 points)

Implement the rgb_decomposition, in which the input image is decomposed into the three channels: R, G and B and return the image excluding the specified channel.

```
In [11]: without_red = rgb_decomposition(load(image1_path), 'R')
         without_blue = rgb_decomposition(load(image1_path), 'B')
         without_green =rgb_decomposition(load(image1_path), 'G')

         display(without_red)
         display(without_blue)
         display(without_green)
```

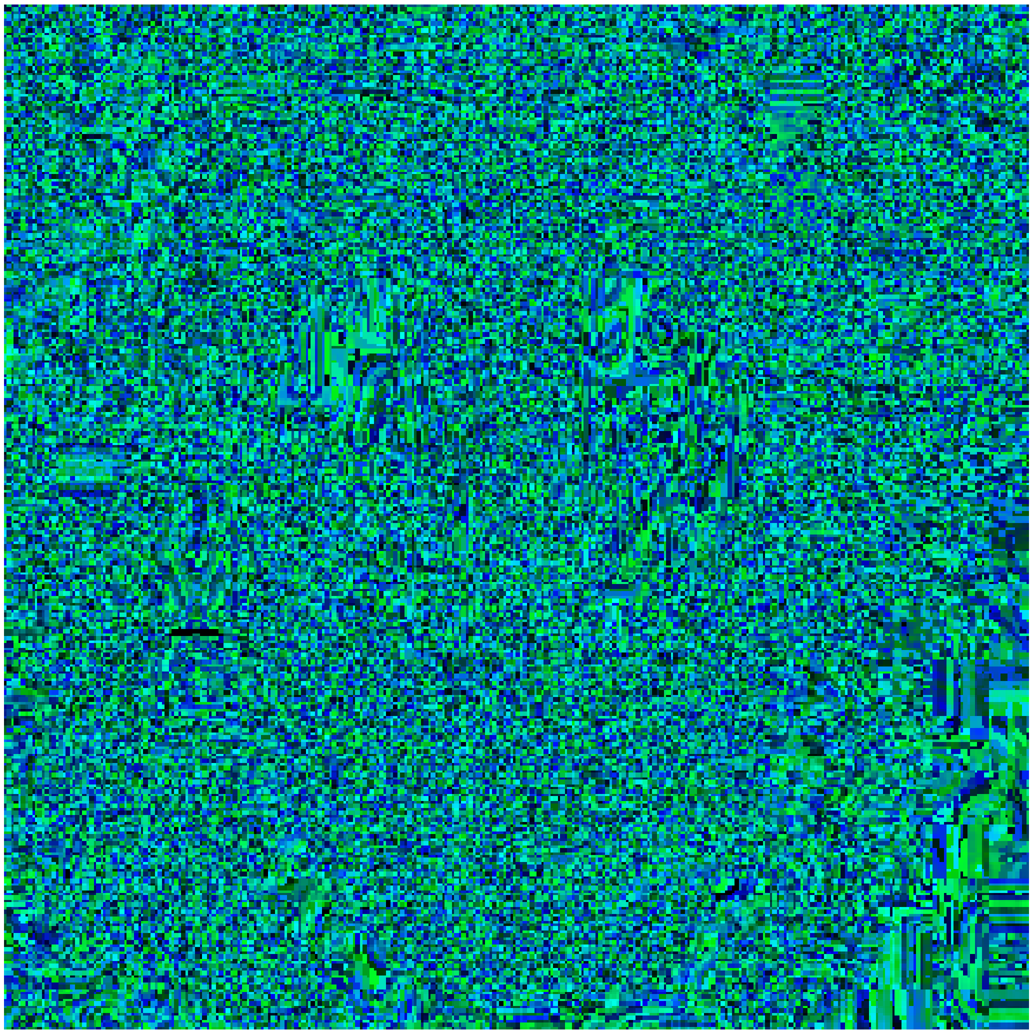


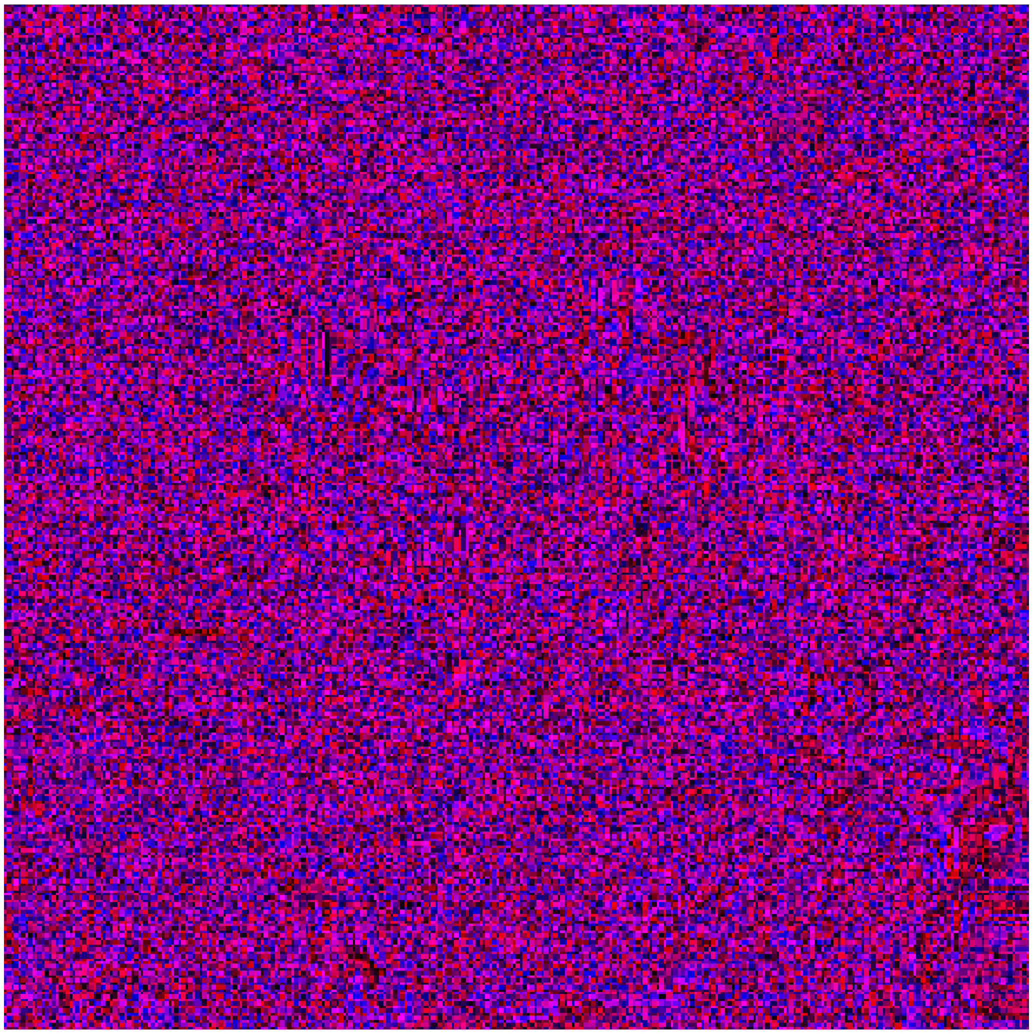
3.5 Question 2.5 (10 points)

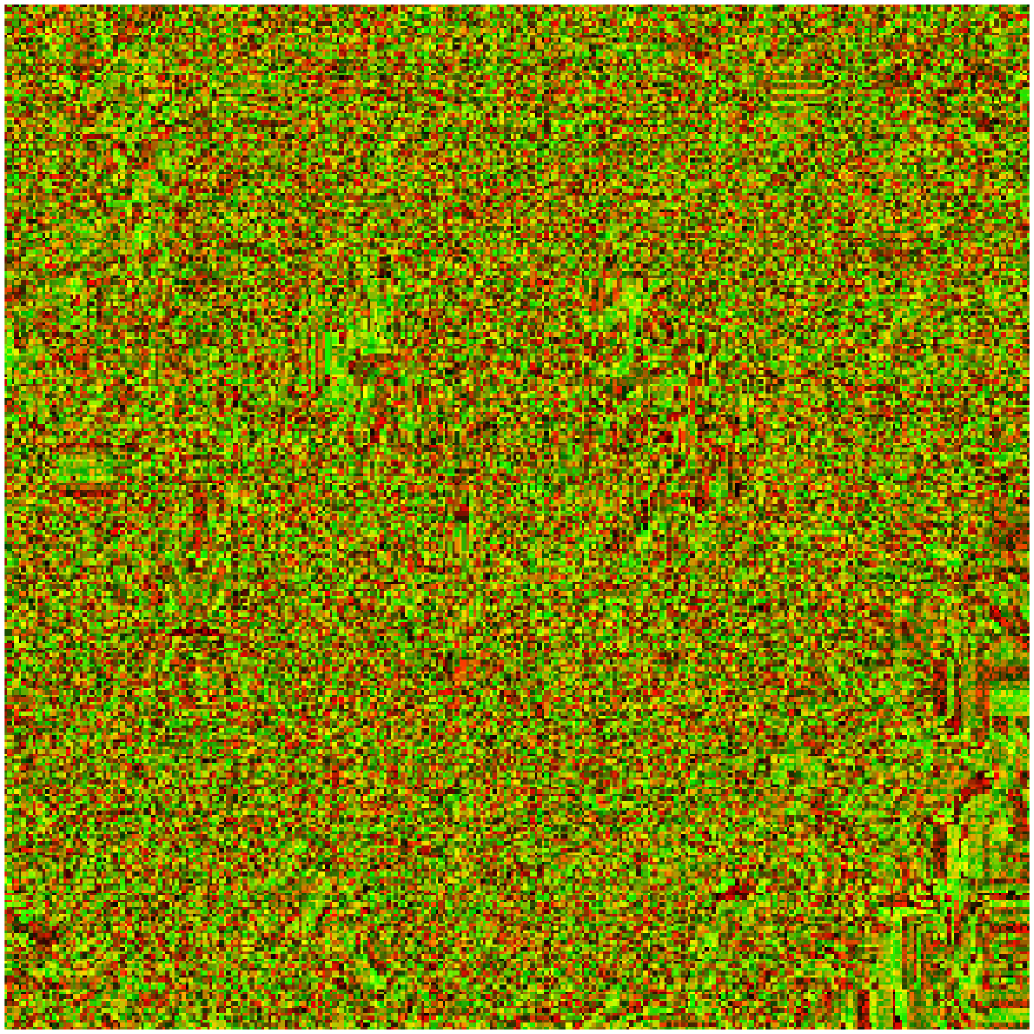
Implement the lab_decomposition, in which the input image is decomposed into the three channels: L, A and B and return the values for the specified channel.

```
In [19]: image_l = lab_decomposition(load(image1_path), 'L')
         image_a = lab_decomposition(load(image1_path), 'A')
         image_b = lab_decomposition(load(image1_path), 'B')

         display(image_l)
         display(image_a)
         display(image_b)
```





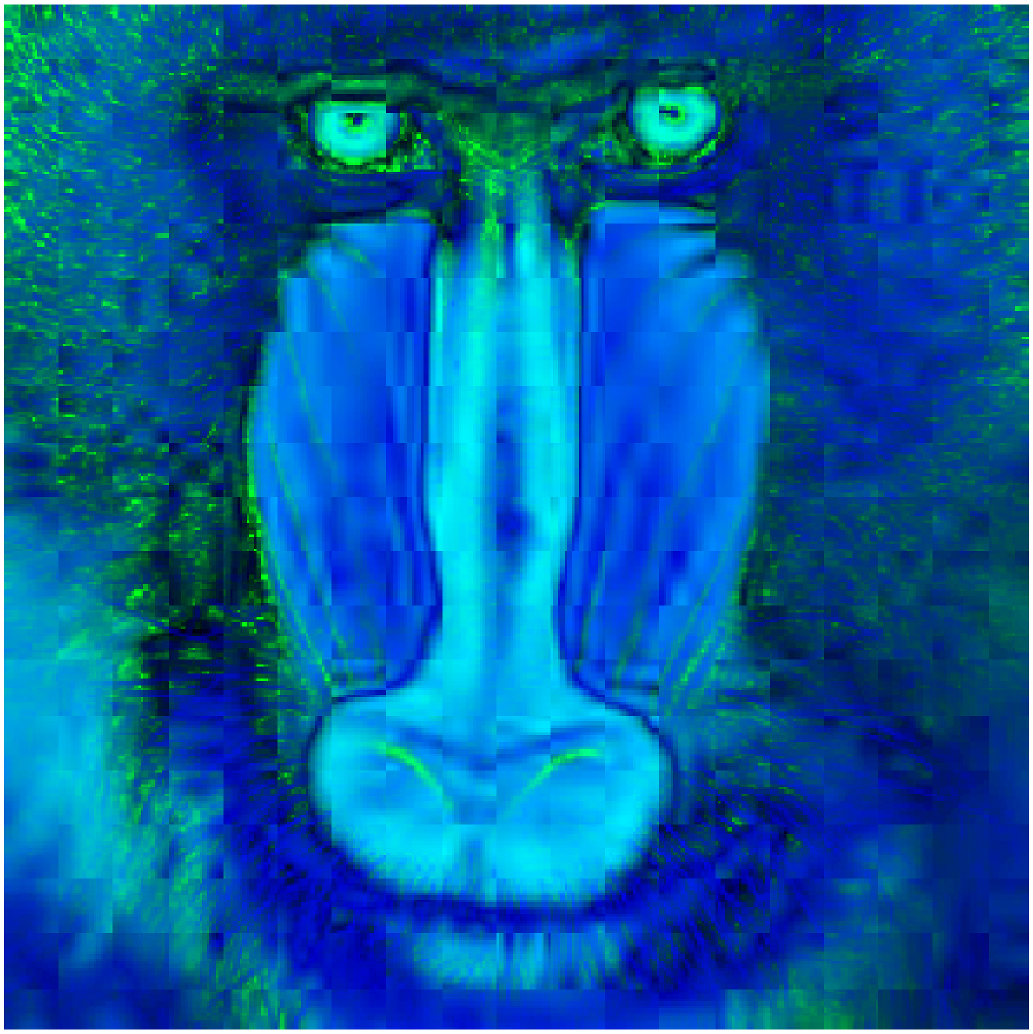


3.6 Question 2.6 (10 points)

Implement the hsv_decomposition, in which the input image is decomposed into the three channels: H, S and V and return the values for the specified channel.

```
In [14]: img = load(image1_path)
         image_h = hsv_decomposition(img, 'H')
         image_s = hsv_decomposition(img, 'S')
         image_v = hsv_decomposition(img, 'V')

         display(image_h)
         display(image_s)
         display(image_v)
```



3.7 Question 2.7 (10 points)

In mix_images method, create a new image such that the left half of the image is the left half of image1 and the right half of the image is the right half of image2. If the channels are specified, exclude the specified channel for the given image.

```
In [17]: image_mixed = mix_images(image1, image2, channel1='R', channel2='B')
         display(image_mixed)
```

```
In [ ]:
```