Gradient of Image

import library

In [ ]:

```
import numpy as np
import matplotlib.image as img
import matplotlib.pyplot as plt
from matplotlib import cm
import matplotlib.colors as colors
```

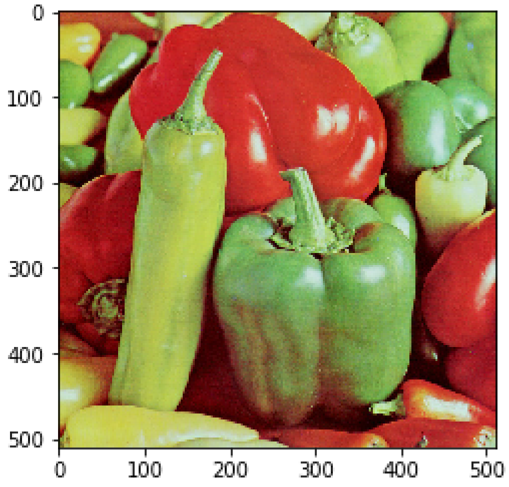
load input image ('test.jpeg')

In [ ]:

```
I0 = img.imread('test.jpeg')
```

In [ ]:

```
plt.imshow(I0)
plt.show()
```



In [ ]:

```
I0
```

Out[ ]:

```
array([[[192,  97,  53],
        [183,  87,  45],
        [189,  94,  48],
        ...,
        [142, 194,  84],
        [111, 195,  84],
        [  1,   0,   8]],

       [[190,  94,  52],
        [190,  95,  51],
        [180,  84,  42],
        ...,
        [142, 195,  81],
        [116, 199,  85],
        [  0,   0,   4]],

       [[179,  86,  43],
        [191,  98,  55],
        [179,  87,  40],
        ...,
        [152, 202,  87],
        [115, 195,  80],
        [  3,   1,   4]],

       ...,

       [[134, 139,  55],
        [129, 141,  55],
        [136, 145,  66],
        ...,
        [168, 198, 164],
        [202, 212, 178],
        [  3,   5,  17]],

       [[133, 131,  46],
        [137, 140,  59],
        [123, 133,  47],
        ...,
        [183, 214, 172],
        [194, 187, 141],
        [  0,   1,   0]],

       [[135, 125,  38],
        [131, 124,  54],
        [122, 134,  32],
        ...,
        [173, 203, 165],
        [193, 172, 129],
        [  2,   1,   0]]], dtype=uint8)
```

In [ ]:

```
np.shape(I0)
```

Out[ ]:

```
(510, 512, 3)
```

check the size of the input image

In [ ]:

```
# ++++++++++++++++++++++++++++++++++++++++++++++++++
# complete the blanks
#
num_row     = np.shape(I0)[0]
num_column  = np.shape(I0)[1]
num_channel = np.shape(I0)[2]
#
# ++++++++++++++++++++++++++++++++++++++++++++++++++

print('number of rows of I0 = ', num_row)
print('number of columns of I0 = ', num_column)
print('number of channels of I0 = ', num_channel)
```

```
number of rows of I0 =  510
number of columns of I0 =  512
number of channels of I0 =  3
```

convert the color image into a grey image

- take the average of the input image with 3 channels with respect to the channels into an image with 1 channel

In [ ]:

```
I0[:,:,0]
```

Out[ ]:

```
array([[192, 183, 189, ..., 142, 111,   1],
       [190, 190, 180, ..., 142, 116,   0],
       [179, 191, 179, ..., 152, 115,   3],
       ...,
       [134, 129, 136, ..., 168, 202,   3],
       [133, 137, 123, ..., 183, 194,   0],
       [135, 131, 122, ..., 173, 193,   2]], dtype=uint8)
```

In [ ]:

```
np.shape(I0[:,:,0])
```

Out[ ]:

```
(510, 512)
```

In [ ]:

```
(I0[:, :, 0] + I0[:, :, 1] + I0[:, :, 2]) / 3
```

Out[ ]:

```
array([[28.66666667, 19.66666667, 25.        , ..., 54.66666667,
        44.66666667,  3.        ],
       [26.66666667, 26.66666667, 16.66666667, ..., 54.        ,
        48.        ,  1.33333333],
       [17.33333333, 29.33333333, 16.66666667, ..., 61.66666667,
        44.66666667,  2.66666667],
       ...,
       [24.        , 23.        , 30.33333333, ...,  6.        ,
        26.66666667,  8.33333333],
       [18.        , 26.66666667, 15.66666667, ..., 19.        ,
         3.33333333,  0.33333333],
       [14.        , 17.66666667, 10.66666667, ...,  9.66666667,
        79.33333333,  1.        ]])
```

In [ ]:

```
np.shape((I0[:, :, 0] + I0[:, :, 1] + I0[:, :, 2]) / 3)
```

Out[ ]:

```
(510, 512)
```

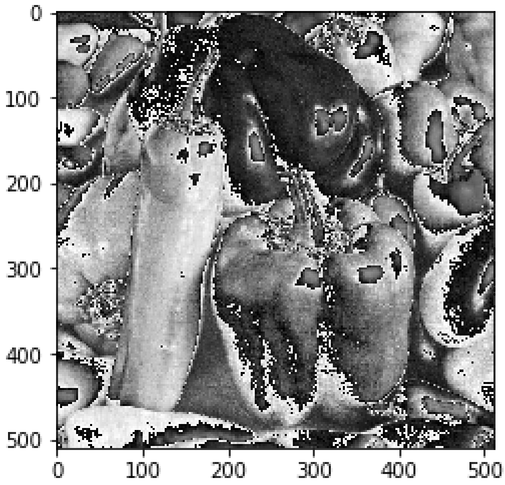
In [ ]:

```
# ++++++++++++++++++++++++++++++++++++++++++++++++++
# complete the blanks
#

I = ( (I0[:, :, 0] + I0[:, :, 1] + I0[:, :, 2]) / 3 )


num_row     = np.shape(I)[0]
num_column  = np.shape(I)[1]
#
# ++++++++++++++++++++++++++++++++++++++++++++++++++

print('number of rows of I = ', num_row)
print('number of columns of I = ', num_column)
```

```
number of rows of I =  510
number of columns of I =  512
```

In [ ]:

```
plt.imshow(I, cmap='gray')
plt.show()
```



normalize the converted image

- normalize the converted grey scale image so that its maximum value is 1 and its minimum value is 0

In [ ]:

```
I
```

Out[ ]:

```
array([[28.66666667, 19.66666667, 25.        , ..., 54.66666667,
        44.66666667,  3.        ],
       [26.66666667, 26.66666667, 16.66666667, ..., 54.        ,
        48.        ,  1.33333333],
       [17.33333333, 29.33333333, 16.66666667, ..., 61.66666667,
        44.66666667,  2.66666667],
       ...,
       [24.        , 23.        , 30.33333333, ...,  6.        ,
        26.66666667,  8.33333333],
       [18.        , 26.66666667, 15.66666667, ..., 19.        ,
         3.33333333,  0.33333333],
       [14.        , 17.66666667, 10.66666667, ...,  9.66666667,
        79.33333333,  1.        ]])
```

In [ ]:

```
# ++++++++++++++++++++++++++++++++++++++++++++++++++
# complete the blanks
#

I = (I - I.min()) / (I.max() - I.min())

#
# ++++++++++++++++++++++++++++++++++++++++++++++++++

print('maximum value of I = ', np.max(I))
print('minimum value of I = ', np.min(I))
```

```
maximum value of I =  1.0
minimum value of I =  0.0
```

In [ ]:

```
I
```

Out[ ]:

```
array([[0.3372549 , 0.23137255, 0.29411765, ..., 0.64313725, 0.5254902 ,
        0.03529412],
       [0.31372549, 0.31372549, 0.19607843, ..., 0.63529412, 0.56470588,
        0.01568627],
       [0.20392157, 0.34509804, 0.19607843, ..., 0.7254902 , 0.5254902 ,
        0.03137255],
       ...,
       [0.28235294, 0.27058824, 0.35686275, ..., 0.07058824, 0.31372549,
        0.09803922],
       [0.21176471, 0.31372549, 0.18431373, ..., 0.22352941, 0.03921569,
        0.00392157],
       [0.16470588, 0.20784314, 0.1254902 , ..., 0.11372549, 0.93333333,
        0.01176471]])
```

In [ ]:

```
np.shape(I)
```

Out[ ]:

```
(510, 512)
```

define a function to compute the derivative of input matrix in x(row)-direction

- forward difference : $I[x+1, y] - I[x, y]$

In [ ]:

```
I
```

Out[ ]:

```
array([[0.3372549 , 0.23137255, 0.29411765, ..., 0.64313725, 0.5254902 ,
        0.03529412],
       [0.31372549, 0.31372549, 0.19607843, ..., 0.63529412, 0.56470588,
        0.01568627],
       [0.20392157, 0.34509804, 0.19607843, ..., 0.7254902 , 0.5254902 ,
        0.03137255],
       ...,
       [0.28235294, 0.27058824, 0.35686275, ..., 0.07058824, 0.31372549,
        0.09803922],
       [0.21176471, 0.31372549, 0.18431373, ..., 0.22352941, 0.03921569,
        0.00392157],
       [0.16470588, 0.20784314, 0.1254902 , ..., 0.11372549, 0.93333333,
        0.01176471]])
```

In [ ]:

```
I[num_row-1]
```

Out[ ]:

```
array([0.16470588, 0.20784314, 0.1254902 , 0.1372549 , 0.04313725,
       0.94509804, 0.13333333, 0.32156863, 0.62745098, 0.81176471,
       0.71372549, 0.69803922, 0.64313725, 0.64313725, 0.77647059,
       0.77254902, 0.8627451 , 0.88627451, 0.90980392, 0.95294118,
       0.92941176, 0.88235294, 0.92941176, 0.98823529, 0.89803922,
       0.96470588, 0.94509804, 0.8745098 , 0.8745098 , 0.89411765,
       0.92156863, 0.9372549 , 0.89803922, 0.9372549 , 0.92941176,
       0.95294118, 0.94117647, 0.9254902 , 0.00784314, 0.04705882,
       0.03529412, 0.1254902 , 1.        , 0.04313725, 0.97647059,
       0.92941176, 1.        , 0.94509804, 0.00392157, 0.00784314,
       0.97647059, 0.04705882, 0.97254902, 0.01568627, 0.01568627,
       0.93333333, 1.        , 0.96862745, 0.91764706, 0.98823529,
       0.98431373, 0.97647059, 0.98823529, 0.98823529, 0.00784314,
       0.03529412, 0.94117647, 0.93333333, 0.9254902 , 0.91764706,
       0.99607843, 0.88627451, 0.9254902 , 0.92941176, 0.89803922,
       0.9372549 , 0.92156863, 0.02352941, 0.03137255, 0.97254902,
       0.07058824, 0.08235294, 0.14509804, 0.2627451 , 0.17647059,
       0.14901961, 0.07058824, 0.07058824, 0.10588235, 0.10196078,
       0.10980392, 0.16078431, 0.10980392, 0.14117647, 0.1372549 ,
       0.10980392, 0.15686275, 0.18431373, 0.16862745, 0.15294118,
       0.0745098 , 0.14901961, 0.11372549, 0.02352941, 0.0627451 ,
       0.01176471, 0.03921569, 0.02352941, 0.01176471, 0.02745098,
       0.98823529, 0.94901961, 0.98823529, 0.92941176, 0.94509804,
       0.91764706, 0.8745098 , 0.86666667, 0.84313725, 0.85098039,
       0.85098039, 0.94509804, 0.97254902, 0.9372549 , 0.85098039,
       0.89019608, 0.87058824, 0.79215686, 0.94901961, 0.92156863,
       0.86666667, 0.89019608, 0.80392157, 0.84313725, 0.73333333,
       0.76470588, 0.83529412, 0.70980392, 0.69019608, 0.76078431,
       0.74901961, 0.92941176, 0.89411765, 0.91764706, 0.83137255,
       0.75294118, 0.94509804, 0.79215686, 0.95294118, 0.87058824,
       0.92156863, 0.91764706, 0.91764706, 0.96470588, 0.89411765,
       0.01176471, 0.87843137, 0.03921569, 0.9372549 , 0.03137255,
       0.94901961, 0.8627451 , 0.10980392, 0.97254902, 0.04705882,
       0.94117647, 0.01960784, 0.97647059, 0.99215686, 0.02352941,
       0.07058824, 0.05882353, 0.08627451, 0.10980392, 0.07843137,
       0.1254902 , 0.11372549, 0.08627451, 0.12941176, 0.1254902 ,
       0.1372549 , 0.16470588, 0.14509804, 0.19215686, 0.15294118,
       0.15294118, 0.19215686, 0.17254902, 0.19215686, 0.25490196,
       0.22745098, 0.22745098, 0.23137255, 0.25490196, 0.30980392,
       0.23529412, 0.25882353, 0.30980392, 0.23921569, 0.27843137,
       0.2627451 , 0.22745098, 0.27843137, 0.28235294, 0.2745098 ,
       0.25882353, 0.22352941, 0.27058824, 0.18039216, 0.18039216,
       0.20392157, 0.19607843, 0.21568627, 0.18823529, 0.14901961,
       0.23529412, 0.18039216, 0.2745098 , 0.11764706, 0.23921569,
       0.19215686, 0.27058824, 0.17647059, 0.23137255, 0.18039216,
       0.16470588, 0.29803922, 0.11372549, 0.28235294, 0.10196078,
       0.21176471, 0.18431373, 0.12941176, 0.26666667, 0.17254902,
       0.22745098, 0.1372549 , 0.22352941, 0.11372549, 0.18823529,
       0.21960784, 0.18039216, 0.16470588, 0.17254902, 0.20784314,
       0.21176471, 0.19607843, 0.17254902, 0.24313725, 0.28627451,
       0.23921569, 0.20392157, 0.18431373, 0.16862745, 0.18431373,
       0.10980392, 0.13333333, 0.16470588, 0.20392157, 0.08627451,
       0.23921569, 0.05490196, 0.33333333, 0.15686275, 0.1372549 ,
       0.02352941, 0.05490196, 0.09803922, 0.83921569, 0.88627451,
       0.18431373, 0.96862745, 0.75686275, 0.84705882, 0.81568627,
       0.79215686, 0.90980392, 0.85882353, 0.89803922, 0.8627451 ,
       0.90980392, 0.99607843, 1.        , 0.10588235, 0.98431373,
       0.03529412, 0.99607843, 0.09803922, 0.16862745, 0.15686275,
       0.1254902 , 0.19215686, 0.19215686, 0.23921569, 0.2       ,
       0.22352941, 0.18039216, 0.29411765, 0.09019608, 0.24313725,
       0.04313725, 0.14509804, 0.16862745, 0.0627451 , 0.07058824,
       0.08627451, 0.21568627, 0.11764706, 0.14509804, 0.08627451,
       0.09411765, 0.09803922, 0.14117647, 0.15294118, 0.1254902 ,
       0.11372549, 0.0745098 , 0.12941176, 0.15294118, 0.13333333,
       0.12941176, 0.07843137, 0.03921569, 0.05882353, 0.09019608,
       0.07058824, 0.1254902 , 0.04313725, 0.04313725, 0.08235294,
       0.03529412, 0.07843137, 0.03137255, 0.0745098 , 0.08627451,
       0.08627451, 0.16078431, 0.18431373, 0.20392157, 0.1372549 ,
       0.1254902 , 0.14509804, 0.16470588, 0.2745098 , 0.32941176,
       0.34901961, 0.28627451, 0.28627451, 0.34901961, 0.36862745,
       0.39215686, 0.39215686, 0.38431373, 0.53333333, 0.58431373,
       0.67843137, 0.62745098, 0.73333333, 0.6627451 , 0.93333333,
       0.00392157, 0.05098039, 0.32156863, 0.24705882, 0.27058824,
       0.34509804, 0.95294118, 0.78431373, 0.68235294, 0.02745098,
       0.09411765, 0.77254902, 0.59215686, 0.75686275, 0.81960784,
       0.9254902 , 0.91764706, 0.90588235, 0.97647059, 0.77254902,
       0.88235294, 0.72941176, 0.64705882, 0.36078431, 0.43137255,
       0.19607843, 0.84313725, 0.90588235, 0.82745098, 0.87058824,
       0.94117647, 0.89019608, 0.91372549, 0.8627451 , 0.96078431,
       0.85490196, 0.95686275, 0.02352941, 0.90196078, 0.94117647,
       0.96862745, 0.02745098, 0.88627451, 0.03137255, 0.03529412,
       0.93333333, 0.00392157, 0.03921569, 0.0745098 , 0.85490196,
       0.04313725, 0.97647059, 0.98823529, 0.90980392, 0.58431373,
       0.66666667, 0.81176471, 0.47843137, 0.4745098 , 0.9372549 ,
       0.77254902, 0.1372549 , 0.09803922, 0.96078431, 0.72941176,
       0.46666667, 0.79607843, 0.71372549, 0.18039216, 0.99607843,
       0.98823529, 0.08627451, 0.98431373, 0.01176471, 0.03529412,
       0.04313725, 0.09019608, 0.00392157, 0.03137255, 0.01176471,
       0.99215686, 1.        , 0.01176471, 0.02352941, 0.01176471,
       1.        , 0.99215686, 0.03137255, 0.        , 0.98039216,
       0.97647059, 0.10196078, 0.01176471, 0.09803922, 0.02352941,
       0.13333333, 0.10588235, 0.0745098 , 0.15686275, 0.09019608,
       0.06666667, 0.10980392, 0.13333333, 0.08627451, 0.07058824,
       0.09411765, 0.15686275, 0.12156863, 0.15686275, 0.0627451 ,
       0.14117647, 0.15294118, 0.22745098, 0.30196078, 0.34509804,
       0.25098039, 0.22352941, 0.29803922, 0.31764706, 0.20392157,
       0.36078431, 0.27843137, 0.52941176, 0.54117647, 0.63921569,
       0.72941176, 0.68627451, 0.75686275, 0.82745098, 0.03921569,
       0.48627451, 0.6       , 0.42352941, 0.27058824, 0.83921569,
       0.69803922, 0.16078431, 0.98039216, 0.0627451 , 0.20392157,
       0.30196078, 0.32941176, 0.25490196, 0.26666667, 0.28627451,
       0.2745098 , 0.28235294, 0.0745098 , 0.05490196, 0.11372549,
       0.93333333, 0.01176471])
```

In [ ]:

```
temp = I.copy()
temp[0] = temp[num_row-1]
```

In [ ]:

```
temp
```

Out[ ]:

```
array([[0.16470588, 0.20784314, 0.1254902 , ..., 0.11372549, 0.93333333,
         0.01176471],
        [0.31372549, 0.31372549, 0.19607843, ..., 0.63529412, 0.56470588,
         0.01568627],
        [0.20392157, 0.34509804, 0.19607843, ..., 0.7254902 , 0.5254902 ,
         0.03137255],
        ...,
        [0.28235294, 0.27058824, 0.35686275, ..., 0.07058824, 0.31372549,
         0.09803922],
        [0.21176471, 0.31372549, 0.18431373, ..., 0.22352941, 0.03921569,
         0.00392157],
        [0.16470588, 0.20784314, 0.1254902 , ..., 0.11372549, 0.93333333,
         0.01176471]])
```

In [ ]:

```
np.roll(temp, -1, axis =0 )
```

Out[ ]:

```
array([[0.31372549, 0.31372549, 0.19607843, ..., 0.63529412, 0.56470588,
         0.01568627],
        [0.20392157, 0.34509804, 0.19607843, ..., 0.7254902 , 0.5254902 ,
         0.03137255],
        [0.2745098 , 0.42352941, 0.27058824, ..., 0.65882353, 0.60784314,
         0.01176471],
        ...,
        [0.21176471, 0.31372549, 0.18431373, ..., 0.22352941, 0.03921569,
         0.00392157],
        [0.16470588, 0.20784314, 0.1254902 , ..., 0.11372549, 0.93333333,
         0.01176471],
        [0.16470588, 0.20784314, 0.1254902 , ..., 0.11372549, 0.93333333,
         0.01176471]])
```

In [ ]:

```
I
```

Out[ ]:

```
array([[0.3372549 , 0.23137255, 0.29411765, ..., 0.64313725, 0.5254902 ,
         0.03529412],
        [0.31372549, 0.31372549, 0.19607843, ..., 0.63529412, 0.56470588,
         0.01568627],
        [0.20392157, 0.34509804, 0.19607843, ..., 0.7254902 , 0.5254902 ,
         0.03137255],
        ...,
        [0.28235294, 0.27058824, 0.35686275, ..., 0.07058824, 0.31372549,
         0.09803922],
        [0.21176471, 0.31372549, 0.18431373, ..., 0.22352941, 0.03921569,
         0.00392157],
        [0.16470588, 0.20784314, 0.1254902 , ..., 0.11372549, 0.93333333,
         0.01176471]])
```

In [ ]:

```
def compute_derivative_x_forward(I):

    D = np.zeros(I.shape)

    # ++++++++++++++++++++++++++++++++++++++++++++++++++
    # complete the blanks
    #
    forward_x = I.copy()
    forward_x[0, :] = forward_x[num_row-1, :]
    forward_x = np.roll(forward_x, -1, axis =0 )
    D = forward_x - I

    #
    # ++++++++++++++++++++++++++++++++++++++++++++++++++

    return D
```

In [ ]:

```
compute_derivative_x_forward(I)
```

Out[ ]:

```
array([[-0.02352941,  0.08235294, -0.09803922, ..., -0.00784314,
         0.03921569, -0.01960784],
       [-0.10980392,  0.03137255,  0.        , ...,  0.09019608,
        -0.03921569,  0.01568627],
       [ 0.07058824,  0.07843137,  0.0745098 , ..., -0.06666667,
         0.08235294, -0.01960784],
       ...,
       [-0.07058824,  0.04313725, -0.17254902, ...,  0.15294118,
        -0.2745098 , -0.09411765],
       [-0.04705882, -0.10588235, -0.05882353, ..., -0.10980392,
         0.89411765,  0.00784314],
       [ 0.        ,  0.        ,  0.        , ...,  0.        ,
         0.        ,  0.        ]])
```

- backward difference : $I[x, y] - I[x - 1, y]$

In [ ]:

```
temp = I.copy()
temp[num_row-1] = temp[0]
temp
```

Out[ ]:

```
array([[0.3372549 , 0.23137255, 0.29411765, ..., 0.64313725, 0.5254902 ,
         0.03529412],
       [0.31372549, 0.31372549, 0.19607843, ..., 0.63529412, 0.56470588,
         0.01568627],
       [0.20392157, 0.34509804, 0.19607843, ..., 0.7254902 , 0.5254902 ,
         0.03137255],
       ...,
       [0.28235294, 0.27058824, 0.35686275, ..., 0.07058824, 0.31372549,
         0.09803922],
       [0.21176471, 0.31372549, 0.18431373, ..., 0.22352941, 0.03921569,
         0.00392157],
       [0.3372549 , 0.23137255, 0.29411765, ..., 0.64313725, 0.5254902 ,
         0.03529412]])
```

In [ ]:

```
np.roll(temp, 1, axis =0 )
```

Out[ ]:

```
array([[0.3372549 , 0.23137255, 0.29411765, ..., 0.64313725, 0.5254902 ,
         0.03529412],
       [0.3372549 , 0.23137255, 0.29411765, ..., 0.64313725, 0.5254902 ,
         0.03529412],
       [0.31372549, 0.31372549, 0.19607843, ..., 0.63529412, 0.56470588,
         0.01568627],
       ...,
       [0.23137255, 0.25882353, 0.30588235, ..., 0.17254902, 0.14509804,
         0.02745098],
       [0.28235294, 0.27058824, 0.35686275, ..., 0.07058824, 0.31372549,
         0.09803922],
       [0.21176471, 0.31372549, 0.18431373, ..., 0.22352941, 0.03921569,
         0.00392157]])
```

In [ ]:

```
I
```

Out[ ]:

```
array([[0.3372549 , 0.23137255, 0.29411765, ..., 0.64313725, 0.5254902 ,
         0.03529412],
       [0.31372549, 0.31372549, 0.19607843, ..., 0.63529412, 0.56470588,
         0.01568627],
       [0.20392157, 0.34509804, 0.19607843, ..., 0.7254902 , 0.5254902 ,
         0.03137255],
       ...,
       [0.28235294, 0.27058824, 0.35686275, ..., 0.07058824, 0.31372549,
         0.09803922],
       [0.21176471, 0.31372549, 0.18431373, ..., 0.22352941, 0.03921569,
         0.00392157],
       [0.16470588, 0.20784314, 0.1254902 , ..., 0.11372549, 0.93333333,
         0.01176471]])
```

In [ ]:

```
def compute_derivative_x_backward(I):

    D = np.zeros(I.shape)

    # ++++++++++++++++++++++++++++++++++++++++++++++++++
    # complete the blanks
    #
    backward_x = I.copy()
    backward_x[num_row-1, :] = backward_x[0, :]
    backward_x = np.roll(backward_x, 1, axis =0 )
    D = I - backward_x


    #
    # ++++++++++++++++++++++++++++++++++++++++++++++++++

    return D
```

- central difference : $\frac{1}{2}(I[x+1,y]-I[x-1,y])$

In [ ]:

```
def compute_derivative_x_central(I):

    D = np.zeros(I.shape)

    # ++++++++++++++++++++++++++++++++++++++++++++++++++
    # complete the blanks
    #
    forward_x = I.copy()
    forward_x[0, :] = forward_x[num_row-1, :]
    forward_x = np.roll(forward_x, -1, axis =0 )
    backward_x = I.copy()
    backward_x[num_row-1, :] = backward_x[0, :]
    backward_x = np.roll(backward_x, 1, axis =0 )
    D = (forward_x - backward_x) / 2

    #
    # ++++++++++++++++++++++++++++++++++++++++++++++++++

    return D
```

define a function to compute the derivative of input matrix in y(column)-direction

- forward difference : $I[x,y+1]-I[x,y]$

In [ ]:

```
np.roll(I, -1, axis =1 )
```

Out[ ]:

```
array([[0.23137255, 0.29411765, 0.31372549, ..., 0.5254902 , 0.03529412,
        0.3372549 ],
       [0.31372549, 0.19607843, 0.25882353, ..., 0.56470588, 0.01568627,
        0.31372549],
       [0.34509804, 0.19607843, 0.28627451, ..., 0.5254902 , 0.03137255,
        0.20392157],
       ...,
       [0.27058824, 0.35686275, 0.05490196, ..., 0.31372549, 0.09803922,
        0.28235294],
       [0.31372549, 0.18431373, 0.16470588, ..., 0.03921569, 0.00392157,
        0.21176471],
       [0.20784314, 0.1254902 , 0.1372549 , ..., 0.93333333, 0.01176471,
        0.16470588]])
```

In [ ]:

```
I
```

Out[ ]:

```
array([[0.3372549 , 0.23137255, 0.29411765, ..., 0.64313725, 0.5254902 ,
        0.03529412],
       [0.31372549, 0.31372549, 0.19607843, ..., 0.63529412, 0.56470588,
        0.01568627],
       [0.20392157, 0.34509804, 0.19607843, ..., 0.7254902 , 0.5254902 ,
        0.03137255],
       ...,
       [0.28235294, 0.27058824, 0.35686275, ..., 0.07058824, 0.31372549,
        0.09803922],
       [0.21176471, 0.31372549, 0.18431373, ..., 0.22352941, 0.03921569,
        0.00392157],
       [0.16470588, 0.20784314, 0.1254902 , ..., 0.11372549, 0.93333333,
        0.01176471]])
```

In [ ]:

```
temp = I.copy()
temp[:,0] = temp[:, num_column-1]
np.roll(temp, -1, axis =1 )
```

Out[ ]:

```
array([[0.23137255, 0.29411765, 0.31372549, ..., 0.5254902 , 0.03529412,
        0.03529412],
       [0.31372549, 0.19607843, 0.25882353, ..., 0.56470588, 0.01568627,
        0.01568627],
       [0.34509804, 0.19607843, 0.28627451, ..., 0.5254902 , 0.03137255,
        0.03137255],
       ...,
       [0.27058824, 0.35686275, 0.05490196, ..., 0.31372549, 0.09803922,
        0.09803922],
       [0.31372549, 0.18431373, 0.16470588, ..., 0.03921569, 0.00392157,
        0.00392157],
       [0.20784314, 0.1254902 , 0.1372549 , ..., 0.93333333, 0.01176471,
        0.01176471]])
```

In [ ]:

```
I
```

Out[ ]:

```
array([[0.3372549 , 0.23137255, 0.29411765, ..., 0.64313725, 0.5254902 ,
        0.03529412],
       [0.31372549, 0.31372549, 0.19607843, ..., 0.63529412, 0.56470588,
        0.01568627],
       [0.20392157, 0.34509804, 0.19607843, ..., 0.7254902 , 0.5254902 ,
        0.03137255],
       ...,
       [0.28235294, 0.27058824, 0.35686275, ..., 0.07058824, 0.31372549,
        0.09803922],
       [0.21176471, 0.31372549, 0.18431373, ..., 0.22352941, 0.03921569,
        0.00392157],
       [0.16470588, 0.20784314, 0.1254902 , ..., 0.11372549, 0.93333333,
        0.01176471]])
```

In [ ]:

```
def compute_derivative_y_forward(I):

    D = np.zeros(I.shape)

    # ++++++++++++++++++++++++++++++++++++++++++++++++++
    # complete the blanks
    #
    forward_y = I.copy()
    forward_y[:,0] = forward_y[:, num_column-1]
    forward_y = np.roll(forward_y, -1, axis =1 )
    D = forward_y - I

    #
    # ++++++++++++++++++++++++++++++++++++++++++++++++++

    return D
```

- backward difference : $I[x, y] - I[x, y-1]$

In [ ]:

```
def compute_derivative_y_backward(I):

    D = np.zeros(I.shape)

    # ++++++++++++++++++++++++++++++++++++++++++++++++++
    # complete the blanks
    #
    backward_y = I.copy()
    backward_y[:, num_column-1] = backward_y[:, 0]
    backward_y = np.roll(backward_y, 1, axis =1 )
    D = I - backward_y


    #
    # ++++++++++++++++++++++++++++++++++++++++++++++++++

    return D
```

- central difference : $\frac{1}{2}(I[x, y+1] - I[x, y-1])$

In [ ]:

```
def compute_derivative_y_central(I):

    D = np.zeros(I.shape)

    # ++++++++++++++++++++++++++++++++++++++++++++++++++
    # complete the blanks
    #
    forward_y = I.copy()
    forward_y[:,0] = forward_y[:, num_column-1]
    forward_y = np.roll(forward_y, -1, axis =1 )
    backward_y = I.copy()
    backward_y[:, num_column-1] = backward_y[:, 0]
    backward_y = np.roll(backward_y, 1, axis =1 )
    D = (forward_y - backward_y) / 2

    #
    # ++++++++++++++++++++++++++++++++++++++++++++++++++

    return D
```

compute the norm of the gradient of the input image

- L_2^2-norm of the gradient $\left(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right)$ is defined by $\left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2$

In [ ]:

```
def compute_norm_gradient_central(I):

    norm_gradient = np.zeros(I.shape)

    # ++++++++++++++++++++++++++++++++++++++++++++++++++
    # complete the blanks
    #
    central_x = compute_derivative_x_central(I)
    central_y = compute_derivative_y_central(I)
    norm_gradient = central_x**2 + central_y**2


    #
    # ++++++++++++++++++++++++++++++++++++++++++++++++++

    return norm_gradient
```

functions for presenting the results

In [ ]:

```
def function_result_01():

    plt.figure(figsize=(8,6))
    plt.imshow(I0)
    plt.show()
```

In [ ]:

```
def function_result_02():

    plt.figure(figsize=(8,6))
    plt.imshow(I, cmap='gray', vmin=0, vmax=1, interpolation='none')
    plt.show()
```

In [ ]:

```
def function_result_03():

    D = compute_derivative_x_forward(I)

    plt.figure(figsize=(8,6))
    plt.imshow(D, cmap='gray')
    plt.show()
```

In [ ]:

```
def function_result_04():

    D = compute_derivative_x_backward(I)

    plt.figure(figsize=(8,6))
    plt.imshow(D, cmap='gray')
    plt.show()
```

In [ ]:

```
def function_result_05():

    D = compute_derivative_x_central(I)

    plt.figure(figsize=(8,6))
    plt.imshow(D, cmap='gray')
    plt.show()
```

In [ ]:

```
def function_result_06():

    D = compute_derivative_y_forward(I)

    plt.figure(figsize=(8,6))
    plt.imshow(D, cmap='gray')
    plt.show()
```

In [ ]:

```
def function_result_07():

    D = compute_derivative_y_backward(I)

    plt.figure(figsize=(8,6))
    plt.imshow(D, cmap='gray')
    plt.show()
```

In [ ]:

```
def function_result_08():

    D = compute_derivative_y_central(I)

    plt.figure(figsize=(8,6))
    plt.imshow(D, cmap='gray')
    plt.show()
```

In [ ]:

```
def function_result_09():

    D = compute_norm_gradient_central(I)

    plt.figure(figsize=(8,6))
    plt.imshow(D, cmap='gray')
    plt.show()
```

In [ ]:

```
def function_result_10():

    D = compute_norm_gradient_central(I)

    plt.figure(figsize=(8,6))
    im = plt.imshow(D, cmap=cm.jet, norm=colors.LogNorm())
    plt.colorbar(im)
    plt.show()
```

In [ ]:

```
def function_result_11():

    D = compute_derivative_x_forward(I)

    value1 = D[0, 0]
    value2 = D[-1, -1]
    value3 = D[100, 100]
    value4 = D[200, 200]

    print('value1 = ', value1)
    print('value2 = ', value2)
    print('value3 = ', value3)
    print('value4 = ', value4)
```

In [ ]:

```
def function_result_12():

    D = compute_derivative_x_backward(I)

    value1 = D[0, 0]
    value2 = D[-1, -1]
    value3 = D[100, 100]
    value4 = D[200, 200]

    print('value1 = ', value1)
    print('value2 = ', value2)
    print('value3 = ', value3)
    print('value4 = ', value4)
```

In [ ]:

```
def function_result_13():

    D = compute_derivative_x_central(I)

    value1 = D[0, 0]
    value2 = D[-1, -1]
    value3 = D[100, 100]
    value4 = D[200, 200]

    print('value1 = ', value1)
    print('value2 = ', value2)
    print('value3 = ', value3)
    print('value4 = ', value4)
```

In [ ]:

```
def function_result_14():

    D = compute_derivative_y_forward(I)

    value1 = D[0, 0]
    value2 = D[-1, -1]
    value3 = D[100, 100]
    value4 = D[200, 200]

    print('value1 = ', value1)
    print('value2 = ', value2)
    print('value3 = ', value3)
    print('value4 = ', value4)
```

In [ ]:

```
def function_result_15():

    D = compute_derivative_y_backward(I)

    value1 = D[0, 0]
    value2 = D[-1, -1]
    value3 = D[100, 100]
    value4 = D[200, 200]

    print('value1 = ', value1)
    print('value2 = ', value2)
    print('value3 = ', value3)
    print('value4 = ', value4)
```

In [ ]:

```
def function_result_16():

    D = compute_derivative_y_central(I)

    value1 = D[0, 0]
    value2 = D[-1, -1]
    value3 = D[100, 100]
    value4 = D[200, 200]

    print('value1 = ', value1)
    print('value2 = ', value2)
    print('value3 = ', value3)
    print('value4 = ', value4)
```

In [ ]:

```
def function_result_17():

    D = compute_norm_gradient_central(I)

    value1 = D[0, 0]
    value2 = D[-1, -1]
    value3 = D[100, 100]
    value4 = D[200, 200]

    print('value1 = ', value1)
    print('value2 = ', value2)
    print('value3 = ', value3)
    print('value4 = ', value4)
```

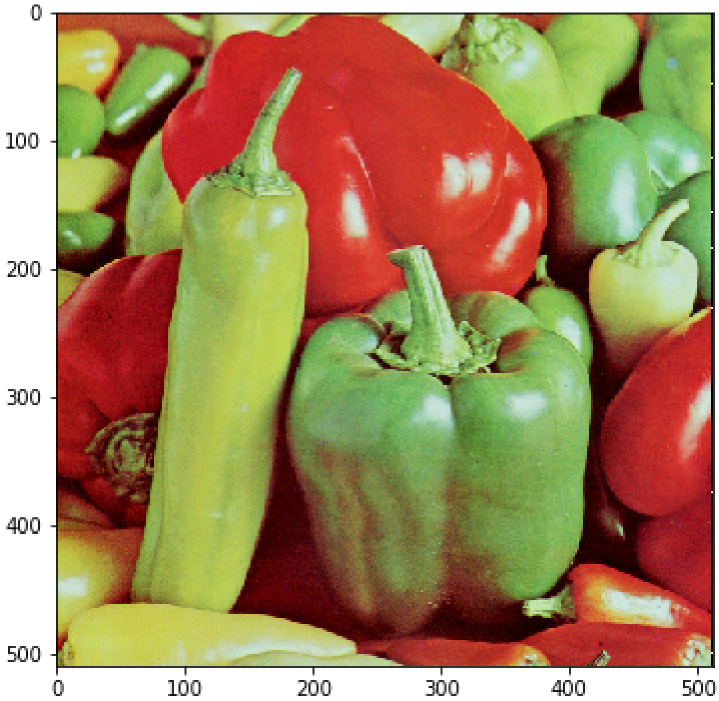
---

results

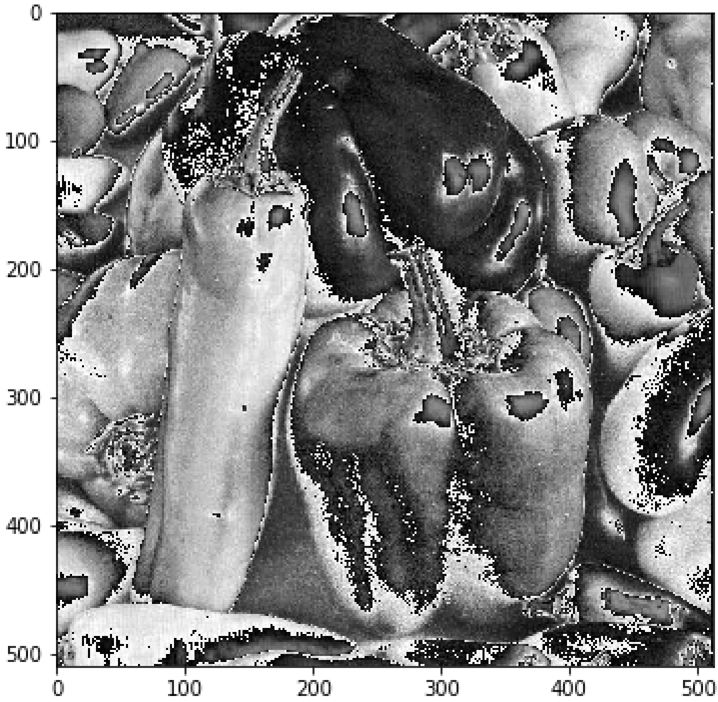
In [ ]:

```
number_result = 17

for i in range(number_result):
    title = '## [RESULT {:02d}]'.format(i+1)
    name_function = 'function_result_{:02d}()'.format(i+1)

    print('**************************************************')
    print(title)
    print('**************************************************')
    eval(name_function)
```

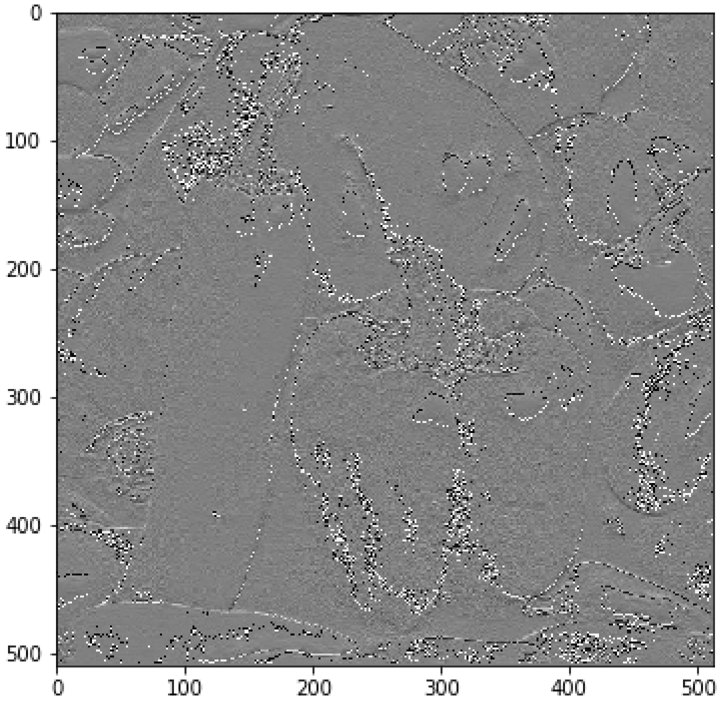
```
**************************************************
## [RESULT 01]
**************************************************
```



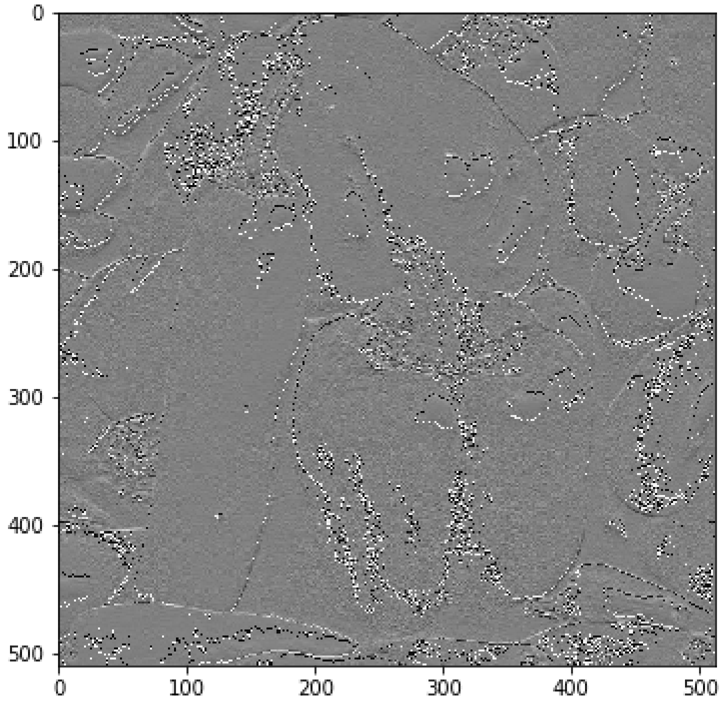
```
**************************************************
## [RESULT 02]
**************************************************
```


0
100
200
300
400
500
0
100
200
300
400
500

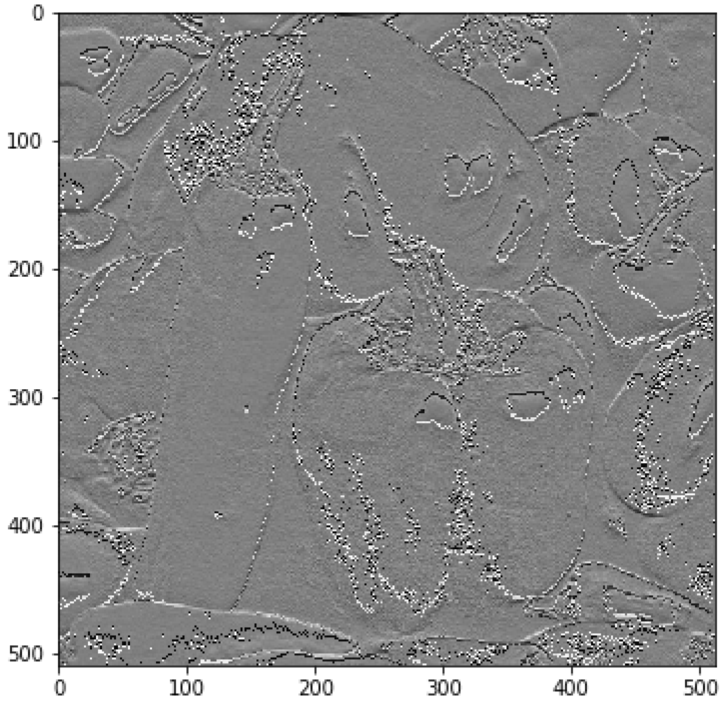
```
*****************************************************
## [RESULT 03]
*****************************************************
```


0
100
200
300
400
500
0
100
200
300
400
500

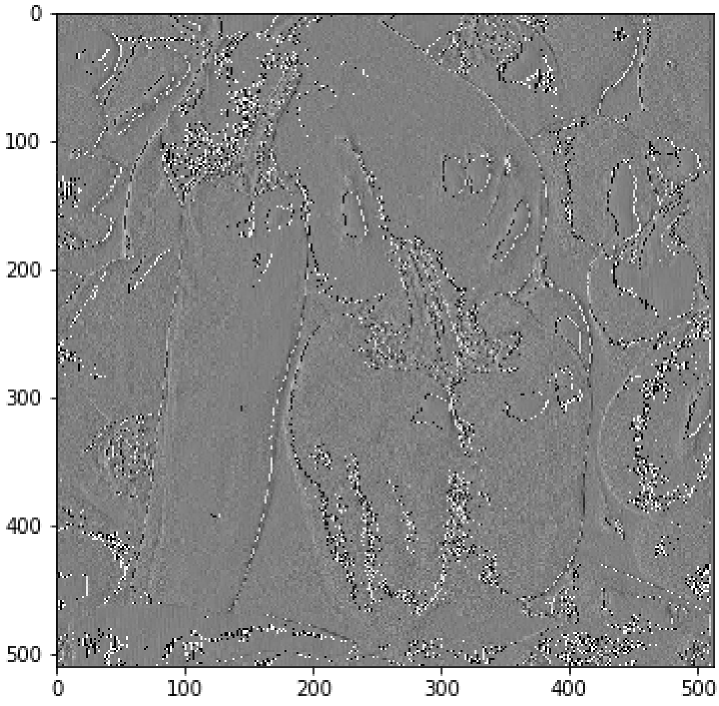
```
**************************************************
## [RESULT 04]
**************************************************
```



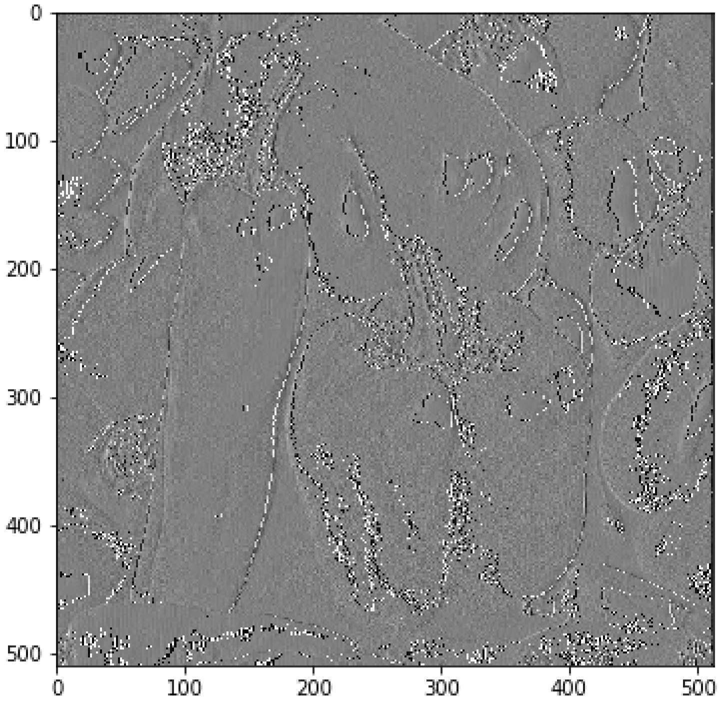
```
**************************************************
## [RESULT 05]
**************************************************
```



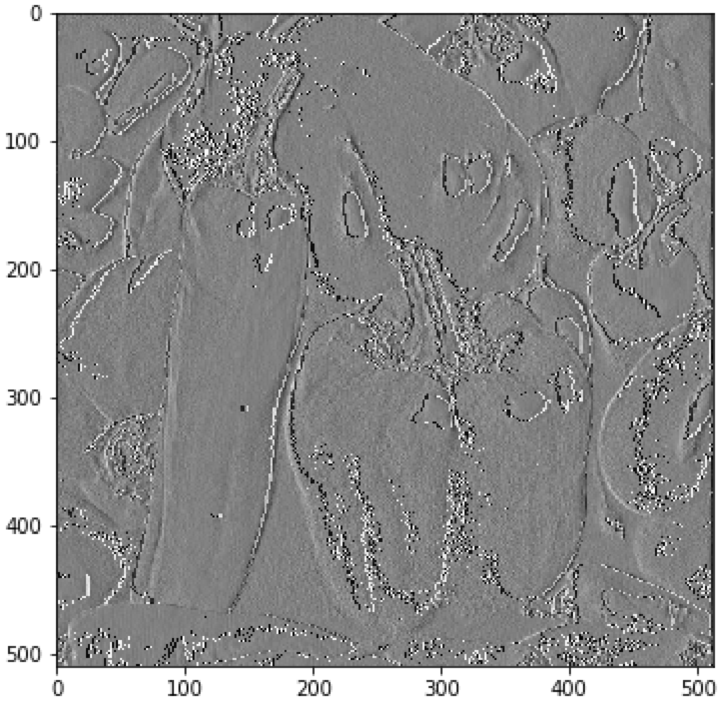
```
**************************************************
## [RESULT 06]
**************************************************
```



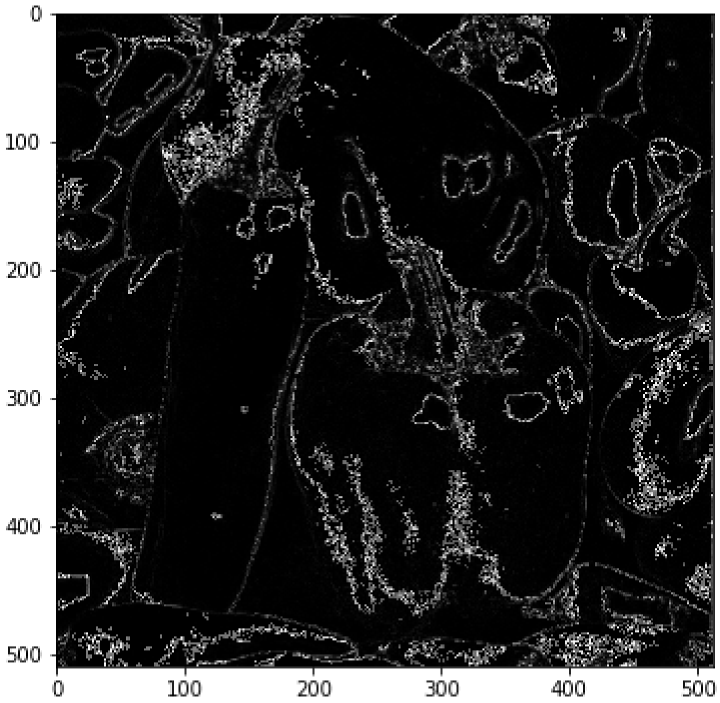
```
**************************************************
## [RESULT 07]
**************************************************
```



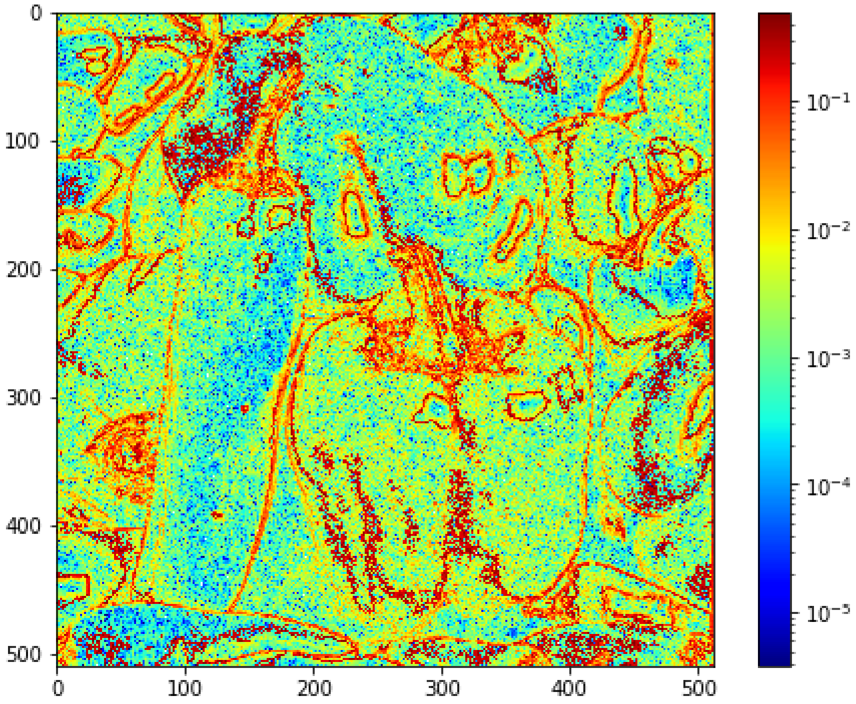
```
**************************************************
## [RESULT 08]
**************************************************
```



```
**************************************************
## [RESULT 09]
**************************************************
```



```
**************************************************
## [RESULT 10]
**************************************************
```


0
100
200
300
400
500
0
100
200
300
400
500
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}

```
**************************************************
## [RESULT 11]
**************************************************
value1 =  -0.023529411764705910
value2 =  0.0
value3 =  -0.01568627450980392
value4 =  0.03529411764705881
**************************************************
## [RESULT 12]
**************************************************
value1 =  0.0
value2 =  0.00784313725490196
value3 =  -0.9568627450980391
value4 =  -0.039215686274509665
**************************************************
## [RESULT 13]
**************************************************
value1 =  -0.011764705882352955
value2 =  0.00392156862745098
value3 =  -0.4862745098039215
value4 =  -0.0019607843137254277
**************************************************
## [RESULT 14]
**************************************************
value1 =  -0.10588235294117648
value2 =  0.0
value3 =  0.9529411764705883
value4 =  0.0
**************************************************
## [RESULT 15]
**************************************************
value1 =  0.0
value2 =  -0.9215686274509803
value3 =  0.027450980392156862
value4 =  0.02352941176470591
**************************************************
## [RESULT 16]
**************************************************
value1 =  -0.05294117647058824
value2 =  -0.46078431372549017
value3 =  0.4901960784313726
value4 =  0.011764705882352955
**************************************************
## [RESULT 17]
**************************************************
value1 =  0.0029411764705882366
value2 =  0.21233756247597074
value3 =  0.47675509419454054
value4 =  0.0001422529796232219
```

In [ ]: