Derivative of Matrix

import library

In [1]:

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

load image

In [2]:

```
I = img.imread('.₩sanfrancisco.jpg')
```

check the size of image

In [3]:

```
n_row = I.shape[0]
n_col = I.shape[1]

print(I.shape)
```

```
(385, 580, 3)
```

convert the input image into gray scale if it is color

In [4]:

```
if I.shape[2] == 3 :

    R = I[:, :, 0]
    G = I[:, :, 1]
    B = I[:, :, 2]
    I = (R+G+B) / 3
    print(I.shape)
```

```
(385, 580)
```

normalize input image so that the range of image is $[0, 1]$

In [5]:

```
I = (I - np.min(I)) / (np.max(I) - np.min(I))
print(I)
```

```
[[0.69803922 0.70588235 0.72941176 ... 0.80784314 0.75294118 0.61176471]
 [0.71764706 0.71764706 0.71764706 ... 0.76078431 0.74117647 0.62352941]
 [0.7372549  0.71764706 0.71764706 ... 0.75294118 0.77647059 0.66666667]
 ...
 [0.17254902 0.1372549  0.16078431 ... 0.95686275 0.02352941 0.91372549]
 [0.15294118 0.14117647 0.14117647 ... 0.96470588 0.03921569 0.91372549]
 [0.16078431 0.19215686 0.1372549  ... 0.91764706 0.89411765 0.87058824]]
```

```
git commit -a -m "load image"
git push origin master
```

generate a matrix to compute the derivative in x-direction

In [6]:

```
Dx = np.eye(n_col, k=-1, dtype=np.float64) - np.eye(n_col,k=0,dtype=np.float64)
```

compute the derivative of I with respect to x-direction

In [7]:

```
Ix = np.dot(I, Dx)
```

```
git commit -a -m "compute the derivative in x-direction"
git push origin master
```

generate a matrix to compute the derivative in y-direction

In [8]:

```
Dy = np.eye(n_row, k=1, dtype=np.float64) - np.eye(n_row,k=0,dtype=np.float64)
```

compute the derivative of I with respect to y-direction

In [9]:

```
Iy = np.dot(Dy, I)
```

```
git commit -a -m "compute the derivative in y-direction"
git push origin master
```

compute L_2^2 of the gradient of I

In [10]:

```
norm_gradient = np.sqrt(Ix**2 + Iy**2)
```

define functions for the visualization

In [11]:

```
def plot_image(I):

    plt.figure(figsize=(10,10))
    plt.imshow(I, norm=colors.LogNorm(), cmap='gray')
    plt.title('input image')
    plt.axis('off')

def plot_image_derivative_x(Ix):

    plt.figure(figsize=(10,10))
    plt.imshow(Ix, norm=colors.LogNorm(), cmap='gray')
    plt.title('image derivative of x')
    plt.axis('off')


def plot_image_derivative_y(Iy):

    plt.figure(figsize=(10,10))
    plt.imshow(Iy, norm=colors.LogNorm(), cmap='gray')
    plt.title('image derivative of y')
    plt.axis('off')


def plot_norm_gradient(norm_gradient):

    plt.figure(figsize=(10,10))
    plt.imshow(norm_gradient, norm=colors.LogNorm(), cmap='gray')
    plt.title('norm gradient')
    plt.axis('off')
```

```
git commit -a -m "define functions for the visualization"
git push origin master
```

##

results

##

01. plot the input image in gray scale

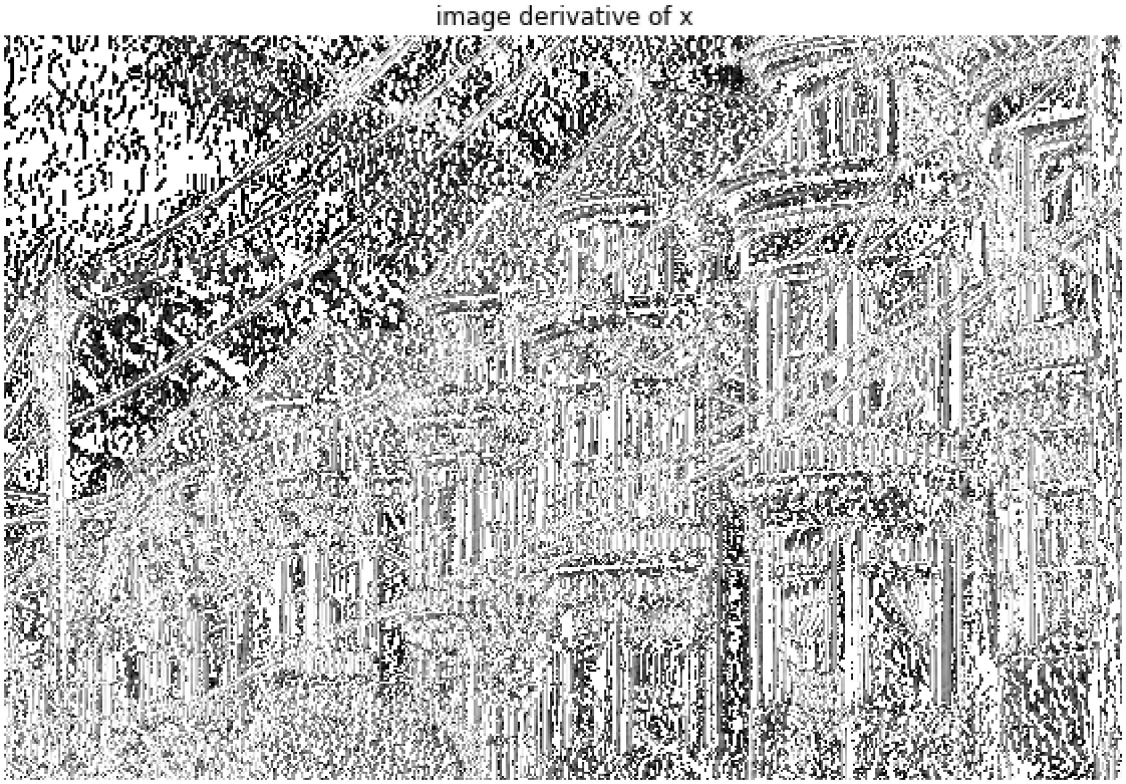
In [12]:

```
plot_image(I)
```

02. plot the derivative I_x of input image in x-direction

In [13]:

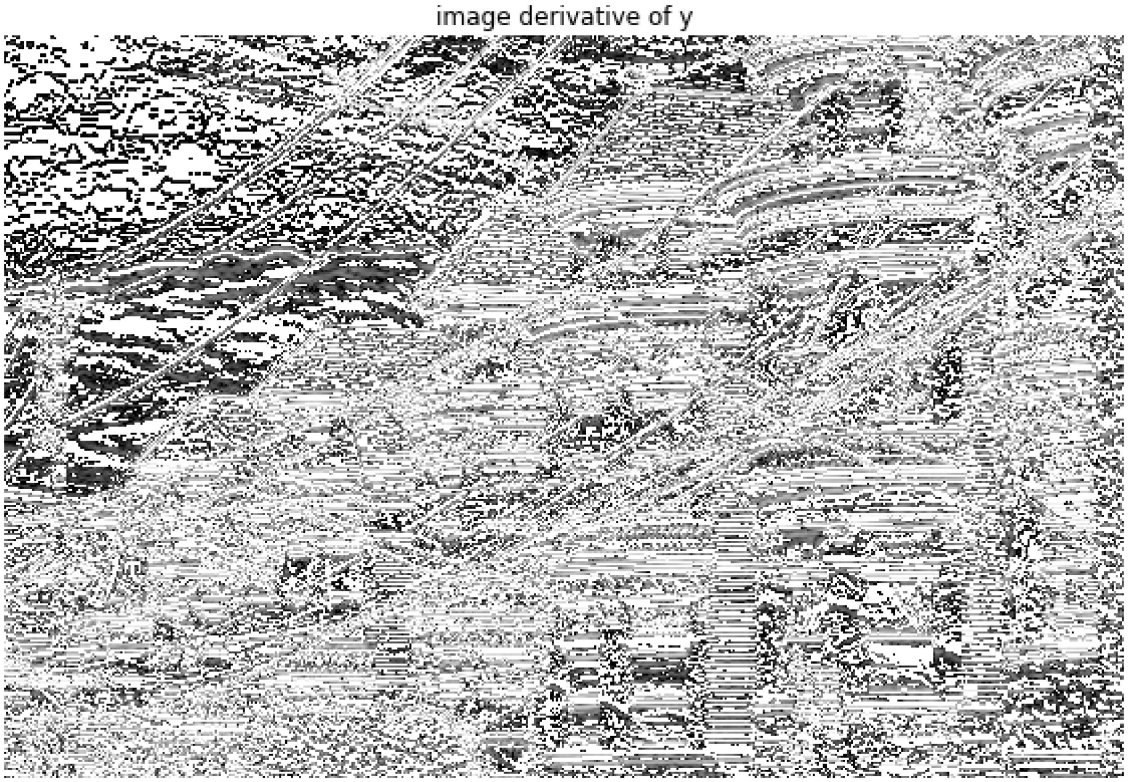
```
plot_image_derivative_x(Ix)
```



03. plot the derivative I_y of input image in y-direction

In [14]:

```
plot_image_derivative_y(Iy)
```



04. plot L_2^2 norm $I_x^2 + I_y^2$ of the gradient of input image

In [15]:

```
plot_norm_gradient(norm_gradient)
```