Assignment05

October 18, 2018

1 Assignment05

1.1 StudentID : 20155212

1.2 Name : Choi Bowon

1.3 GitHub : https://github.com/ChoiBowon/Assignment

1.4 Import packages for project

```
In [286]: import matplotlib.pyplot as plt
          import numpy as np
          from scipy import signal
          from skimage import io, color
          from skimage import exposure
```

1.5 Input file image and transform to gray scale image

```
In [287]: file_image = 'cau.jpg'

          im_color = io.imread(file_image)
          im_gray  = color.rgb2gray(im_color)
```

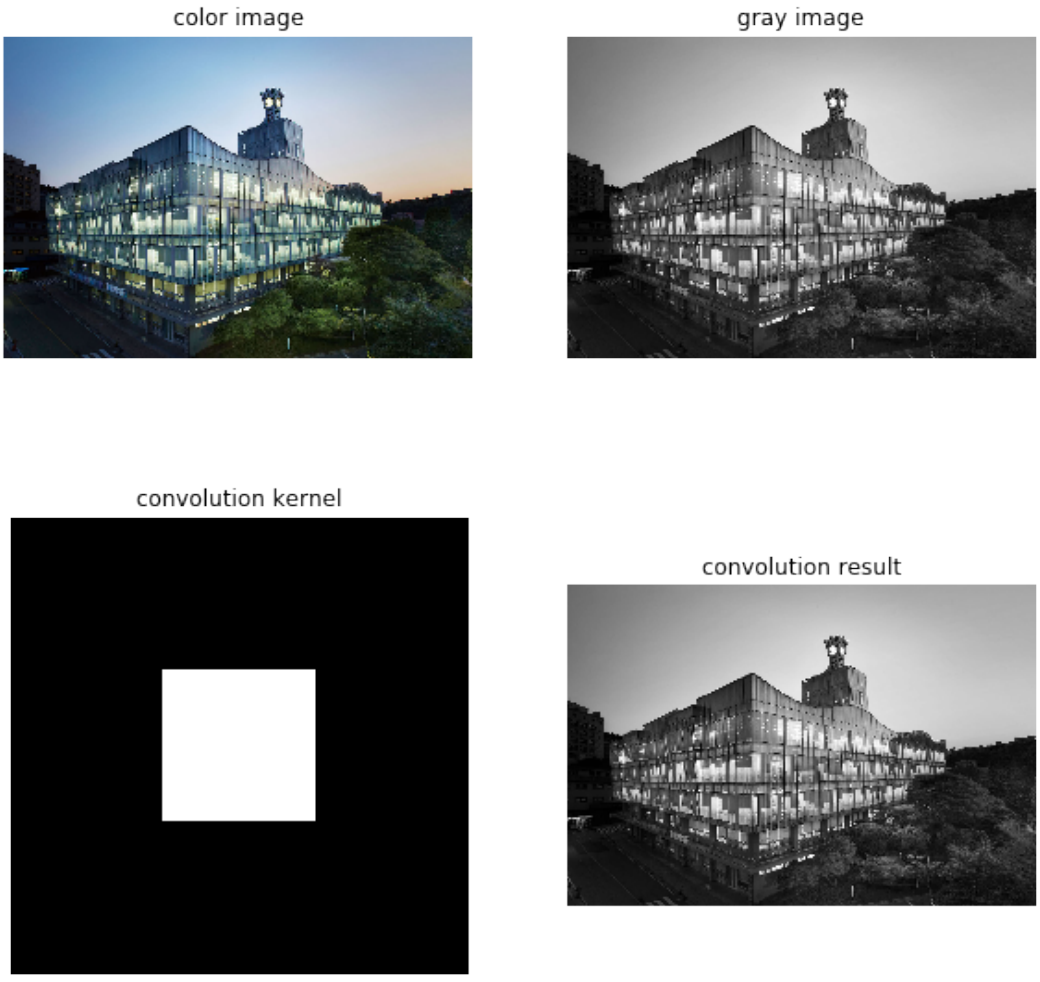
1.6 Define kernels for example

```
In [288]: ker = np.array([[0,0,0],[0,1,0],[0,0,0]])
          im_conv = signal.convolve2d(im_gray, ker, boundary='symm', mode='same')
```

1.7 Plot the images

```
In [289]: plt.figure(figsize=(10,10))
          p1 = plt.subplot(2,2,1)
          p1.set_title('color image')
          plt.imshow(im_color)
          plt.axis('off')

          p2 = plt.subplot(2,2,2)
          p2.set_title('gray image')
          plt.imshow(im_gray, cmap='gray')
```

```
plt.axis('off')

p3 = plt.subplot(2,2,3)
p3.set_title('convolution kernel')
plt.imshow(ker, cmap='gray')
plt.axis('off')

p4 = plt.subplot(2,2,4)
p4.set_title('convolution result')
plt.imshow(im_conv, cmap='gray')
plt.axis('off')

plt.show()
```



1.8 Define kernels for computing image gradients

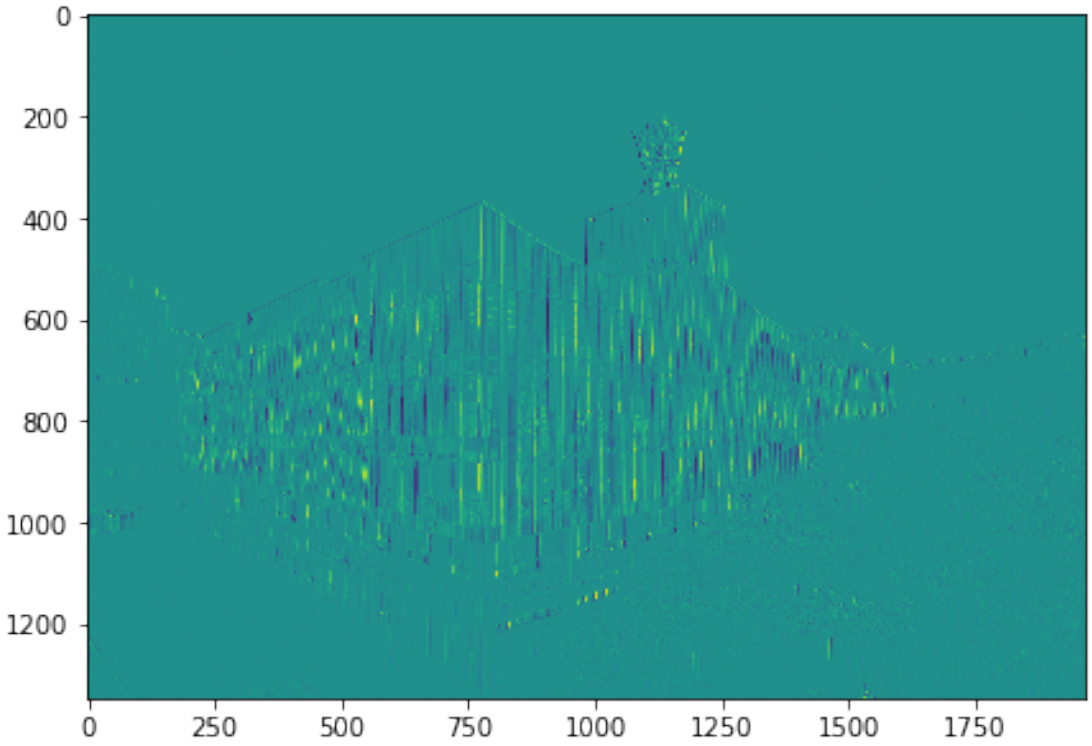
```
In [290]: ker_gradient = np.array([[1,0,-1],[1,0,-1],[1,0,-1]])
```

1.9 Function for computing x_derivative and y_derivative and shows the result

1.9.1 The y_derivative function uses transpose matrix of image

```
In [291]: x_result = x_derivative()
          def x_derivative():
              x_result = signal.convolve2d(im_gray, ker_gradient, boundary='symm', mode='same')
              plt.figure(figsize=(7,7))
              plt.imshow(x_result)
              print("x derivative image")

              return x_result
```

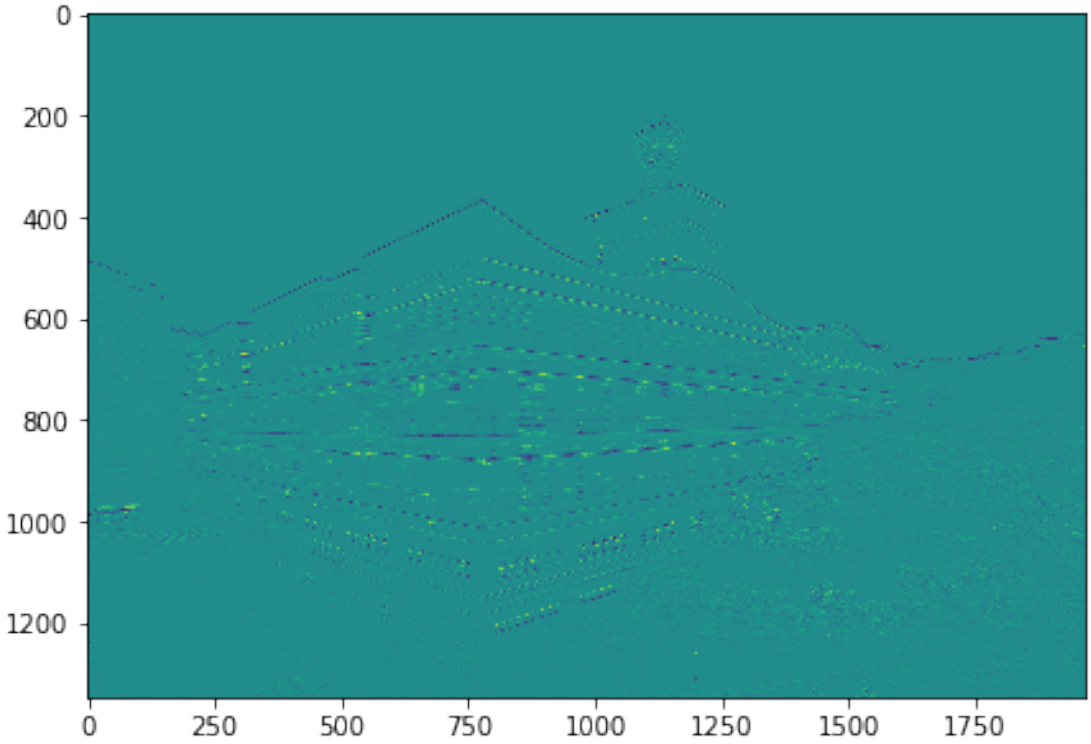
```
x derivative image
```




```
In [292]: def y_derivative():
              y_result = signal.convolve2d(im_gray.T, ker_gradient, boundary='symm', mode='same')
              plt.figure(figsize=(7,7))
              plt.imshow(y_result.T)
```

```
            print("y derivative image")

            return y_result.T
        y_result = y_derivative()
```

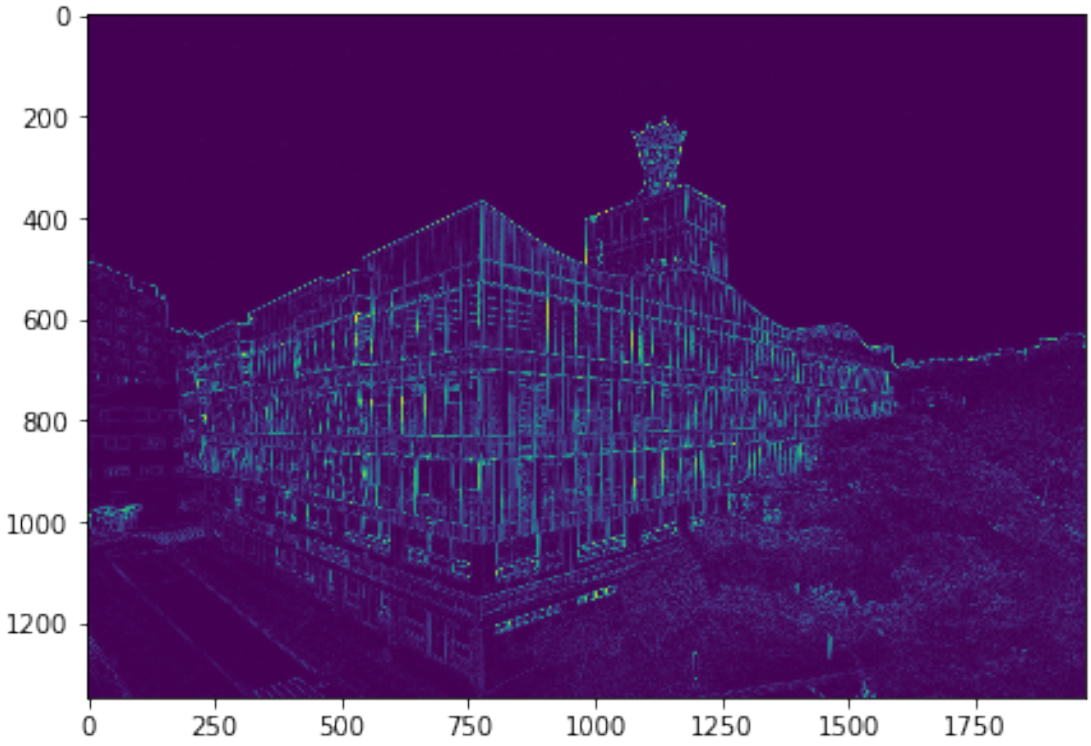
```
y derivative image
```



1.10 Function for computing the magnitude of the gradient and Showing the result images

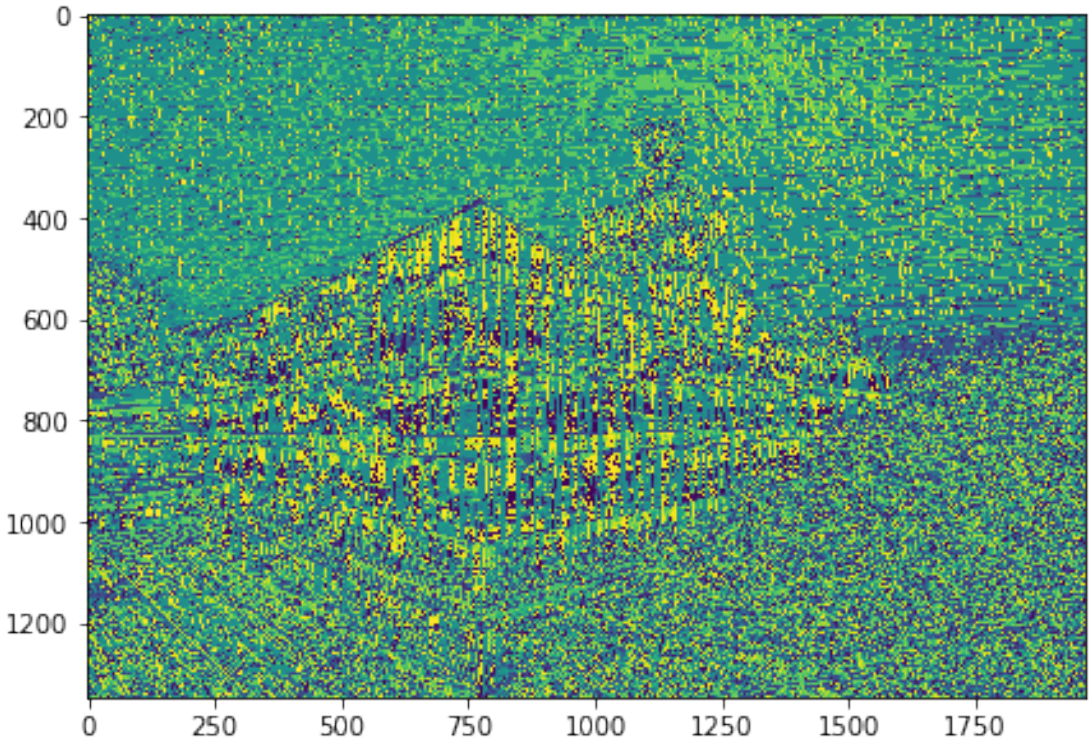
```
In [293]: def magnitude(x, y):
              absoulte = (x**2 + y**2)**0.5
              plt.figure(figsize=(7,7))
              plt.imshow(absoulte)
              print("absoulte image")

In [294]: magnitude(x_result, y_result)
```

```
absoulte image
```



1.11 Function for computing the direction of the gradient using tangent method and Showing the result of direction

```
In [295]: def calculate_angle():
              angle = np.arctan2(y_result, x_result)
              plt.figure(figsize=(7,7))
              plt.imshow(angle)
              print("direction image")

In [296]: calculate_angle()

direction image
```



1.12 Define kernels for smoothing image

```
In [297]: ker_soomth = np.array([[1/3,1/3,1/3],[1/3,1/3,1/3],[1/3,1/3,1/3]])
```

1.13 Define function for smoothing and show the result of smoothing image function

```
In [298]: def smooth():
              s_result = signal.convolve2d(im_gray, ker_soomth, boundary='symm', mode='same')

              plt.figure(figsize=(20,20))
              p5 = plt.subplot(1,2,1)
              p5.set_title('gray image')
              plt.imshow(im_gray, cmap='gray')
              plt.axis('off')

              p6 = plt.subplot(1,2,2)
              p6.set_title('smooth image')
              plt.imshow(s_result, cmap='gray')
              plt.axis('off')

              plt.show()

In [299]: smooth()
```

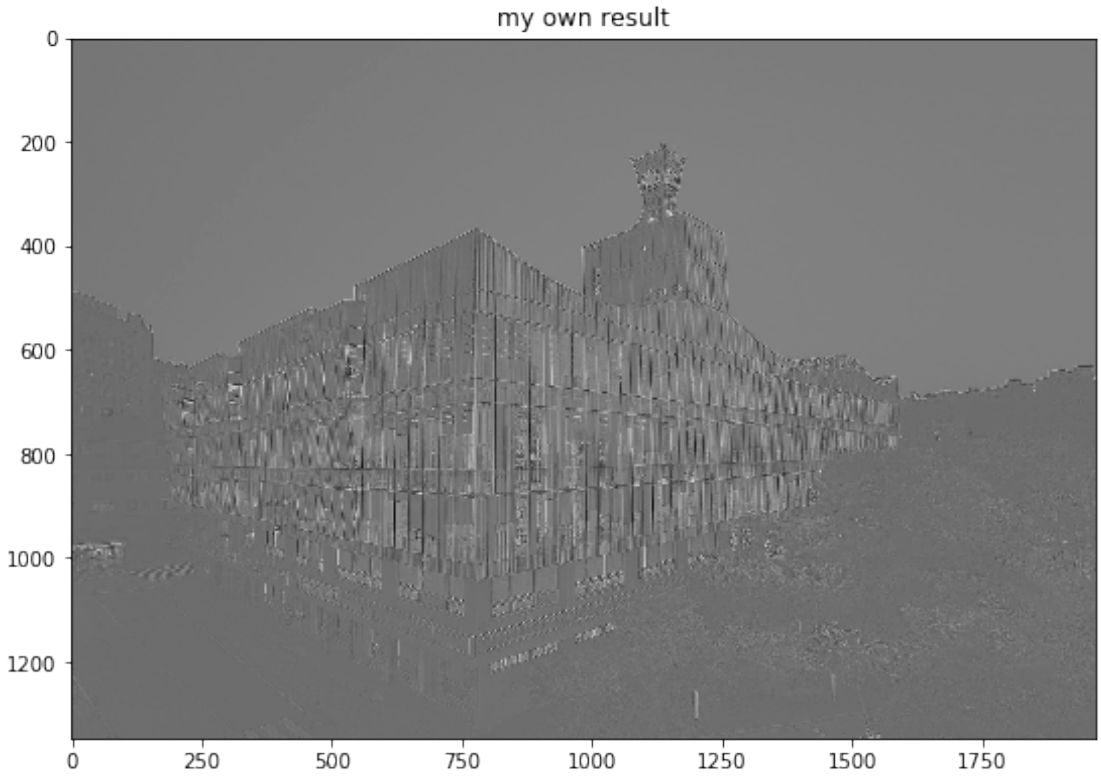
1.14 Define my own kernel

```
In [300]: ker_own = np.array([[-1,-1,-1],[-1,9,-1],[-1,-1,-1]])
```

1.15 Show the result of convolution with my own kernal

1.15.1 It shows the edge of gray scale image

```
In [301]: im_conv_own = signal.convolve2d(im_gray, ker_own, boundary='symm', mode='same')

          plt.figure(figsize=(20,20))
          p7 = plt.subplot(1,2,1)
          p7.set_title('my own result')
          plt.imshow(im_conv_own, cmap='gray')

Out[301]: <matplotlib.image.AxesImage at 0x1c315cc320>
```


my own result
0
200
400
600
800
1000
1200
0
250
500
750
1000
1250
1500
1750