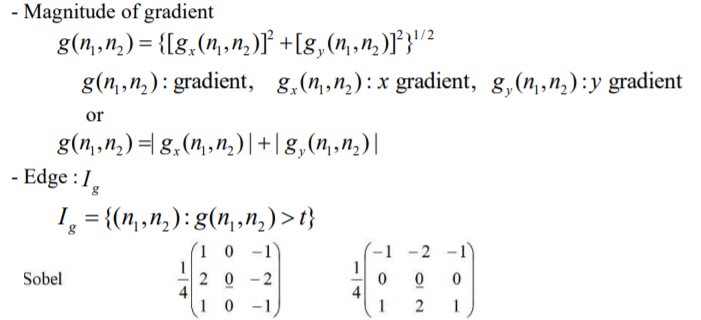
**영상처리 Python 실습 과제 3**

201620350 김지영

Edge detection을 위해서 다음과 같은 python program을 작성하고 실행하여 결과를 관찰하시오.

(a) Edge detection을 하기 위한 python program을 작성하여 lena.png, boats.png 영상에 대해서 실행하고 edge detection을 위한 임계값을 변화시킬 때 결과 영상의 변화를 관찰하시오. (Sobel operator 사용 (x 방향과 y 방향의 sobel operator를 각각 정의하고 ndimage.convolve를 사용하여 x 방향의 gradient와 y 방향의 gradient를 각각 구함)



[그림1 – lena.png]

- Python Code

import numpy as np

from scipy import ndimage

from matplotlib import pyplot as plt

img = plt.imread('lena.png')

img = np.uint8(255\*img)

plt.imshow(img,'gray',vmin=0,vmax=255)

plt.title("Input Image")

plt.show()

# sobel operator

filter\_x = np.array([[-1, 0, 1], [-2, 0, 2], [-1, 0, 1]])

filter\_y = np.array([[-1, -2, -1], [0, 0, 0], [1, 2, 1]])

h, w = img.shape

newImage\_x = np.zeros((h, w))

newImage\_y = np.zeros((h, w))

newgradientImage = np.zeros((h, w))

# define gradient

imgthres = np.zeros([h,w])

for i in range(1, h - 1):

for j in range(1, w - 1):

if(img[i,j]>=255): # 임계값 = 255 인 경우

imgthres[i,j]=255

else:

imgthres[i,j]=0

gradient\_x = (filter\_x[0, 0] \* img[i - 1, j - 1]) + \

(filter\_x[0, 1] \* img[i - 1, j]) + \

(filter\_x[0, 2] \* img[i - 1, j + 1]) + \

(filter\_x[1, 0] \* img[i, j - 1]) + \

(filter\_x[1, 1] \* img[i, j]) + \

(filter\_x[1, 2] \* img[i, j + 1]) + \

(filter\_x[2, 0] \* img[i + 1, j - 1]) + \

(filter\_x[2, 1] \* img[i + 1, j]) + \

(filter\_x[2, 2] \* img[i + 1, j + 1])

newImage\_x[i - 1, j - 1] = ndimage.convolve(img[i - 1, j - 1], gradient\_x)

gradient\_y = (filter\_y[0, 0] \* img[i - 1, j - 1]) + \

(filter\_y[0, 1] \* img[i - 1, j]) + \

(filter\_y[0, 2] \* img[i - 1, j + 1]) + \

(filter\_y[1, 0] \* img[i, j - 1]) + \

(filter\_y[1, 1] \* img[i, j]) + \

(filter\_y[1, 2] \* img[i, j + 1]) + \

(filter\_y[2, 0] \* img[i + 1, j - 1]) + \

(filter\_y[2, 1] \* img[i + 1, j]) + \

(filter\_y[2, 2] \* img[i + 1, j + 1])

newImage\_y[i - 1, j - 1] = ndimage.convolve(img[i - 1, j - 1], gradient\_y)

# Edge Magnitude

magnitude = np.sqrt(pow(gradient\_x, 2.0) + pow(gradient\_y, 2.0))

newgradientImage[i - 1, j - 1] = magnitude

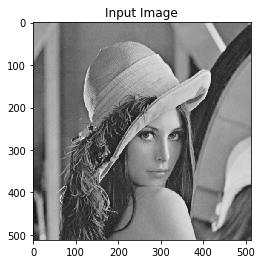
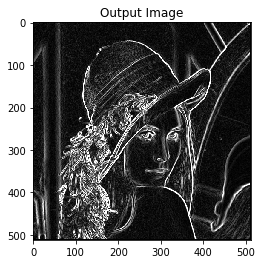
plt.imshow(newgradientImage,'gray',vmin=0,vmax=255)

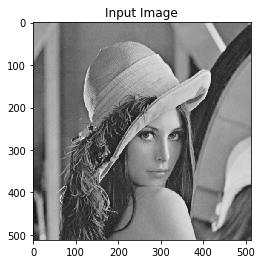
plt.title("Output Image")

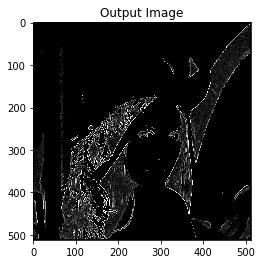
plt.show()

- Image

(1) 임계값 = 255

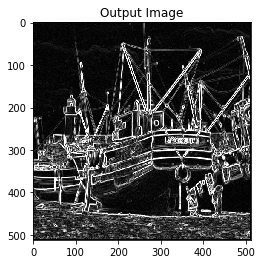
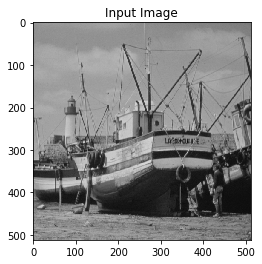
(2) 임계값 = 80



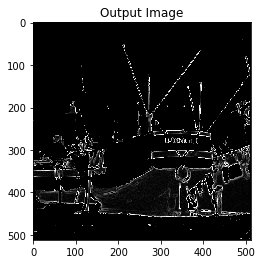
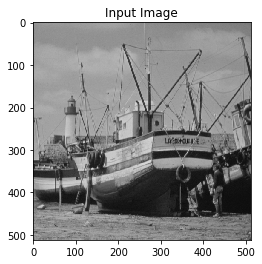
[그림2 – boats.png]

- Image

(1) 임계값 = 255



(2) 임계값 = 80



=> Edge detection을 하기 위한 python program을 lena.png, boats.png 영상에 대해 실행한 결과입니다. 임계값이 커질수록 edge가 더 많이 검출되는 것을 확인하였습니다.

(b) (a)번에서 작성한 program에서 edge detection 부분을 function에서 수행하도록 code를 수정하고 동작을 검증하시오.

- Python Code

import numpy as np

from scipy import ndimage

from matplotlib import pyplot as plt

img = plt.imread('lena.png')

img = np.uint8(255\*img)

plt.imshow(img,'gray',vmin=0,vmax=255)

plt.title("Input Image")

plt.show()

# sobel operator

filter\_x = np.array([[-1, 0, 1], [-2, 0, 2], [-1, 0, 1]])

filter\_y = np.array([[-1, -2, -1], [0, 0, 0], [1, 2, 1]])

h, w = img.shape

newImage\_x = np.zeros((h, w))

newImage\_y = np.zeros((h, w))

newgradientImage = np.zeros((h, w))

# edge detection function

def edge\_fn():

imgthres = np.zeros([h,w])

for i in range(1, h - 1):

for j in range(1, w - 1):

if(img[i,j]>=255): # 임계값 변화시키는 부분

imgthres[i,j]=255

else:

imgthres[i,j]=0

gradient\_x = (filter\_x[0, 0] \* img[i - 1, j - 1]) + \

(filter\_x[0, 1] \* img[i - 1, j]) + \

(filter\_x[0, 2] \* img[i - 1, j + 1]) + \

(filter\_x[1, 0] \* img[i, j - 1]) + \

(filter\_x[1, 1] \* img[i, j]) + \

(filter\_x[1, 2] \* img[i, j + 1]) + \

(filter\_x[2, 0] \* img[i + 1, j - 1]) + \

(filter\_x[2, 1] \* img[i + 1, j]) + \

(filter\_x[2, 2] \* img[i + 1, j + 1])

newImage\_x[i - 1, j - 1] = ndimage.convolve(img[i - 1, j - 1], gradient\_x)

gradient\_y = (filter\_y[0, 0] \* img[i - 1, j - 1]) + \

(filter\_y[0, 1] \* img[i - 1, j]) + \

(filter\_y[0, 2] \* img[i - 1, j + 1]) + \

(filter\_y[1, 0] \* img[i, j - 1]) + \

(filter\_y[1, 1] \* img[i, j]) + \

(filter\_y[1, 2] \* img[i, j + 1]) + \

(filter\_y[2, 0] \* img[i + 1, j - 1]) + \

(filter\_y[2, 1] \* img[i + 1, j]) + \

(filter\_y[2, 2] \* img[i + 1, j + 1])

newImage\_y[i - 1, j - 1] = ndimage.convolve(img[i - 1, j - 1], gradient\_y)

# Edge Magnitude

magnitude = np.sqrt(pow(gradient\_x, 2.0) + pow(gradient\_y, 2.0))

newgradientImage[i - 1, j - 1] = magnitude

return imgthres

edge\_fn()

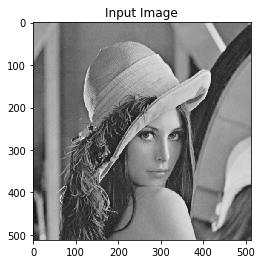
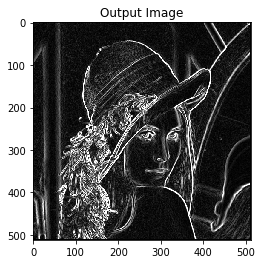
plt.imshow(newgradientImage,'gray',vmin=0,vmax=255)

plt.title("Output Image")

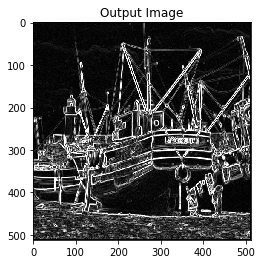
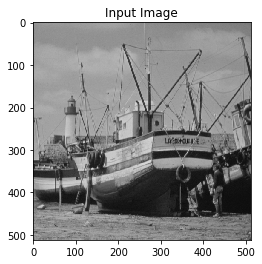
plt.show()

- Image

(1) lena.png

(2) boats.png



=> 실행결과가 (a)와 같습니다.

(c) (b)번에서 작성한 program에서 edge detection을 위한 function을 별도의 edge\_fn.py 분리하여 작성하고 edge detection을 포함하는 file을 module로 import하여 function을 call하는 program으로 수정하여 다시 작성하고(두개의 python code file을 하나의 project로 구성) 실행하여 결과가 (a), (b)와 같은지 검토하시오.

- edge\_fn.py

import numpy as np

from scipy import ndimage

from matplotlib import pyplot as plt

img = plt.imread('lena.png')

img = np.uint8(255\*img)

# sobel operator

filter\_x = np.array([[-1, 0, 1], [-2, 0, 2], [-1, 0, 1]])

filter\_y = np.array([[-1, -2, -1], [0, 0, 0], [1, 2, 1]])

h, w = img.shape

newImage\_x = np.zeros((h, w))

newImage\_y = np.zeros((h, w))

newgradientImage = np.zeros((h, w))

# edge detection function

def ed\_fn():

imgthres = np.zeros([h,w])

for i in range(1, h - 1):

for j in range(1, w - 1):

if(img[i,j]>=255): # 임계값 변화시키는 부분

imgthres[i,j]=255

else:

imgthres[i,j]=0

gradient\_x = (filter\_x[0, 0] \* img[i - 1, j - 1]) + \

(filter\_x[0, 1] \* img[i - 1, j]) + \

(filter\_x[0, 2] \* img[i - 1, j + 1]) + \

(filter\_x[1, 0] \* img[i, j - 1]) + \

(filter\_x[1, 1] \* img[i, j]) + \

(filter\_x[1, 2] \* img[i, j + 1]) + \

(filter\_x[2, 0] \* img[i + 1, j - 1]) + \

(filter\_x[2, 1] \* img[i + 1, j]) + \

(filter\_x[2, 2] \* img[i + 1, j + 1])

newImage\_x[i - 1, j - 1] = ndimage.convolve(img[i - 1, j - 1], gradient\_x)

gradient\_y = (filter\_y[0, 0] \* img[i - 1, j - 1]) + \

(filter\_y[0, 1] \* img[i - 1, j]) + \

(filter\_y[0, 2] \* img[i - 1, j + 1]) + \

(filter\_y[1, 0] \* img[i, j - 1]) + \

(filter\_y[1, 1] \* img[i, j]) + \

(filter\_y[1, 2] \* img[i, j + 1]) + \

(filter\_y[2, 0] \* img[i + 1, j - 1]) + \

(filter\_y[2, 1] \* img[i + 1, j]) + \

(filter\_y[2, 2] \* img[i + 1, j + 1])

newImage\_y[i - 1, j - 1] = ndimage.convolve(img[i - 1, j - 1], gradient\_y)

# Edge Magnitude

magnitude = np.sqrt(pow(gradient\_x, 2.0) + pow(gradient\_y, 2.0))

newgradientImage[i - 1, j - 1] = magnitude

return imgthres

- ed\_dect\_test4.py

import numpy as np

from scipy import ndimage

from matplotlib import pyplot as plt

from sub1 import edge\_fn

img = plt.imread('lena.png')

img = np.uint8(255\*img)

plt.imshow(img,'gray',vmin=0,vmax=255)

plt.title("Input Image")

plt.show()

# sobel operator

filter\_x = np.array([[-1, 0, 1], [-2, 0, 2], [-1, 0, 1]])

filter\_y = np.array([[-1, -2, -1], [0, 0, 0], [1, 2, 1]])

h, w = img.shape

newImage\_x = np.zeros((h, w))

newImage\_y = np.zeros((h, w))

newgradientImage = np.zeros((h, w))

# edge detection function

def ed\_fn():

imgthres = np.zeros([h,w])

for i in range(1, h - 1):

for j in range(1, w - 1):

if(img[i,j]>=255): # 임계값 변화시키는 부분

imgthres[i,j]=255

else:

imgthres[i,j]=0

gradient\_x = (filter\_x[0, 0] \* img[i - 1, j - 1]) + \

(filter\_x[0, 1] \* img[i - 1, j]) + \

(filter\_x[0, 2] \* img[i - 1, j + 1]) + \

(filter\_x[1, 0] \* img[i, j - 1]) + \

(filter\_x[1, 1] \* img[i, j]) + \

(filter\_x[1, 2] \* img[i, j + 1]) + \

(filter\_x[2, 0] \* img[i + 1, j - 1]) + \

(filter\_x[2, 1] \* img[i + 1, j]) + \

(filter\_x[2, 2] \* img[i + 1, j + 1])

newImage\_x[i - 1, j - 1] = ndimage.convolve(img[i - 1, j - 1], gradient\_x)

gradient\_y = (filter\_y[0, 0] \* img[i - 1, j - 1]) + \

(filter\_y[0, 1] \* img[i - 1, j]) + \

(filter\_y[0, 2] \* img[i - 1, j + 1]) + \

(filter\_y[1, 0] \* img[i, j - 1]) + \

(filter\_y[1, 1] \* img[i, j]) + \

(filter\_y[1, 2] \* img[i, j + 1]) + \

(filter\_y[2, 0] \* img[i + 1, j - 1]) + \

(filter\_y[2, 1] \* img[i + 1, j]) + \

(filter\_y[2, 2] \* img[i + 1, j + 1])

newImage\_y[i - 1, j - 1] = ndimage.convolve(img[i - 1, j - 1], gradient\_y)

# Edge Magnitude

magnitude = np.sqrt(pow(gradient\_x, 2.0) + pow(gradient\_y, 2.0))

newgradientImage[i - 1, j - 1] = magnitude

return imgthres

ed\_fn()

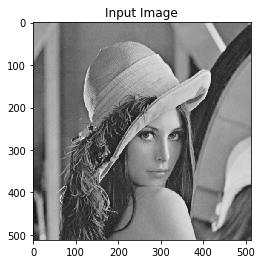
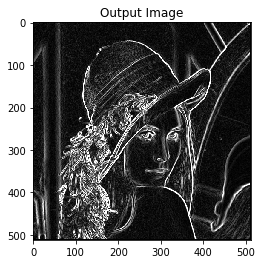
plt.imshow(newgradientImage,'gray',vmin=0,vmax=255)

plt.title("Output Image")

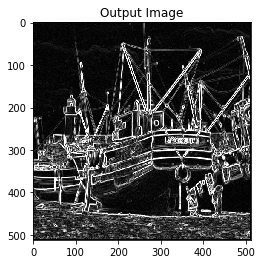
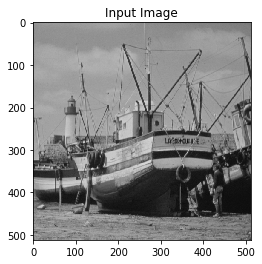
plt.show()

- Image

(1) lena.png

(2) boats.png



=> 실행결과가 (a), (b)와 같습니다.