Spatial Image Enhancement

電機所　609415159　江昀融

Date due:2020/10/28

Date handed in:2020/10/22

Technical description:

使用　opencv 用於圖片的基本讀寫與顯示
numpy 用於進行陣列操作
math 用於影像處理的公式計算
matplotlib 用於值方圖的顯示

```
import cv2 as cv
import numpy as np
import math
from matplotlib import pyplot as plt
```

cv.imread 讀取輸入圖片
並且讀取圖片長寬並顯示

```
#choose read which image
#img = cv.imread('Lena.bmp', cv.IMREAD_GRAYSCALE)
img = cv.imread('Peppers.bmp', cv.IMREAD_GRAYSCALE)
#img = cv.imread('Cameraman.bmp', cv.IMREAD_GRAYSCALE)

#print height and width of input image
height, width = img.shape
print(type(img))
print(img.shape)
print("height = ", height)
print("width = ", width)
```

接下來對三張輸入影像使用三種影像增強技術

(1) power-law (gamma) transformation

先創建空的矩陣空間（p_law）
並且以指數轉換 $s = cr^{\gamma}$
使用 for 迴圈走訪每個像素並做指數計算
將結果存入目標矩陣並以圖像顯示

```
#creat new image for output
p_law = np.zeros( (height, width), np.uint8 )

#power-law (gamma) transformation
# s = cr^y
#recommend y = 0.4 for Lena.bmp
#recommend y = 6   for Peppers.bmp
#recommend y = 1.2 for Cameraman.bmp
c = 255
y = 6

for i in range(height):
    for j in range(width):
        p_law[i][j] = c*((img[i][j]/c)**y)

cv.imshow('input', img)
cv.imshow('power_law', p_law)
```

(2) histogram equalization

```
#creat new image for output and array of cdf & pdf
h_eq = np.zeros( (height, width), np.uint8 )
pdf = np.zeros( (256))
cdf = np.zeros( (256))

#calculation pdf
print("wait for produce pdf ......")
for k in range(256):
    freq = 0
    for i in range(height):
        for j in range(width):
            if img[i][j] == k: freq += 1
    pdf[k] = freq
print("pdf done")

#calculation cdf
for i in range(256):
    for j in range(i+1):
        cdf[i] = cdf[i] + pdf[j]
print("cdf done")
print("min of cdf = ", np.min(cdf))

#histogram equalization
for i in range(height):
    for j in range(width):
        h_eq[i][j] = (cdf[img[i][j]] - np.min(cdf))/((height*width) - np.min(cdf))*(256 - 1)

print("histogram equalization done")
cv.imshow('input', img)
cv.imshow('histogram_equalization', h_eq)
```

先創建空的矩陣空間
（p_law & pdf & cdf）
將每個像素正規劃，並使用公式近似機率公式的微分計算與積分部份

$$p_r(r_k) = \frac{n_k}{n}$$

使用此公式並以 for 迴圈計算並存入 pdf 陣列

$$s_k = T(r_k) = \sum_{j=0}^{k} p_r(r_j)$$

$$= \sum_{j=0}^{k} \frac{n_j}{n} \qquad k = 0, 1, \ldots, L-1.$$

使用此公式以 for 迴圈將 pdf 累加至 cdf 陣列

$$h(v) = \text{round}\left(\frac{cdf(v) - cdf_{min}}{cdf_{max} - cdf_{min}} \times (L-1)\right)$$

最後以此公式計算均勻化，將各像素以本身輸入的像素分佈機率進行重新分佈，達成將集中的像素分散的成果
最後將結果存入矩陣並以圖像顯示

```
#show histogram
plt.subplot(2, 1, 1)
plt.hist(img.ravel(), 256, [0, 255],label= 'original image')
plt.subplot(2, 1, 2)
plt.hist(h_eq.ravel(), 256, [0, 255],label= 'histogram_equalization image')
plt.savefig('histogram.png')
plt.show()
```

使用了 pyplot 的功能計算像素的分佈並繪製值方圖
以兩張圖一起的方式顯示

(3) image sharpening by Laplacian

$$\nabla^2 f = [f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1)] - 4f(x,y).$$

以拉普拉斯方程式推導至此公式
並將圖像的座標矩陣以此算式計算

$$g(x,y) = \begin{cases} f(x,y) - \nabla^2 f(x,y) \\ f(x,y) + \nabla^2 f(x,y) \end{cases}$$

以拉普拉斯公式的結果為強化圖片的特徵
所以要與原圖做相加或相減的計算

```
#creat new image for output & laplacian kernel
lap_s = np.zeros( (height-2, width-2), np.uint8 )
s_kernel = np.zeros((3, 3))
s_kernel[1][1] = 5
s_kernel[0][1] = -1
s_kernel[1][0] = -1
s_kernel[2][1] = -1
s_kernel[1][2] = -1
```

先創建空的矩陣空間（lap_s）
以公式得出的結果創建矩陣

laplacian kernel =

0	−1	0
−1	5	−1
0	−1	0

```
#run convolution by laplacian kernel
print("wait for convolution ......")

for i in range(height-2):
    for j in range(width-2):
        cal = (s_kernel[0][0]*img[i][j] +
               s_kernel[0][1]*img[i][j+1] +
               s_kernel[0][2]*img[i][j+2] +
               s_kernel[1][0]*img[i+1][j] +
               s_kernel[1][1]*img[i+1][j+1] +
               s_kernel[1][2]*img[i+1][j+2] +
               s_kernel[2][0]*img[i+2][j] +
               s_kernel[2][1]*img[i+2][j+1] +
               s_kernel[2][2]*img[i+2][j+2])
        if(cal > 0):lap_s[i][j] = cal
        if(cal >= 255):lap_s[i][j] = 255

print("convolution done")
cv.imshow('input', img)
cv.imshow('laplacian sharpening', lap_s)
```

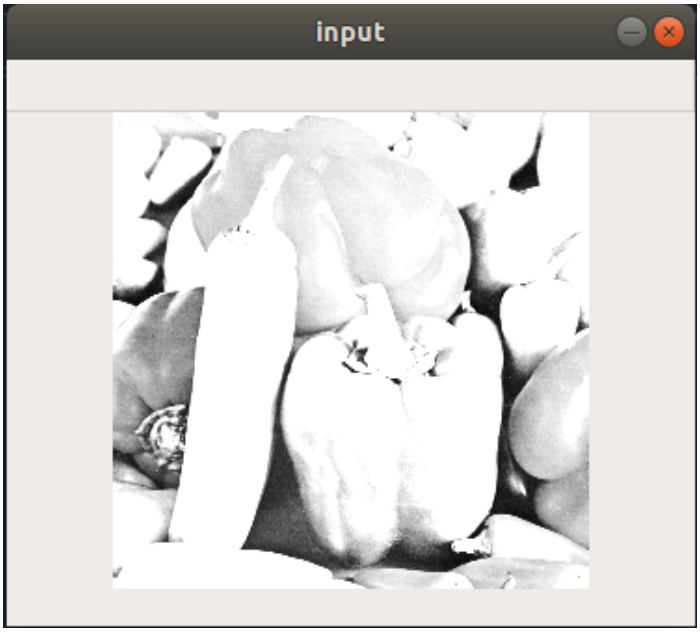
最後針對周圍像素與中心像素做計算，
相當於輸入進行卷積計算
使用 for 迴圈對原圖與矩陣卷積計算
將結果存入目標矩陣
並以圖片顯示

Experimental results:

(1) power-law (gamma) transformation

Lena.bmp y = 0.4

Peppers.bmp y = 6

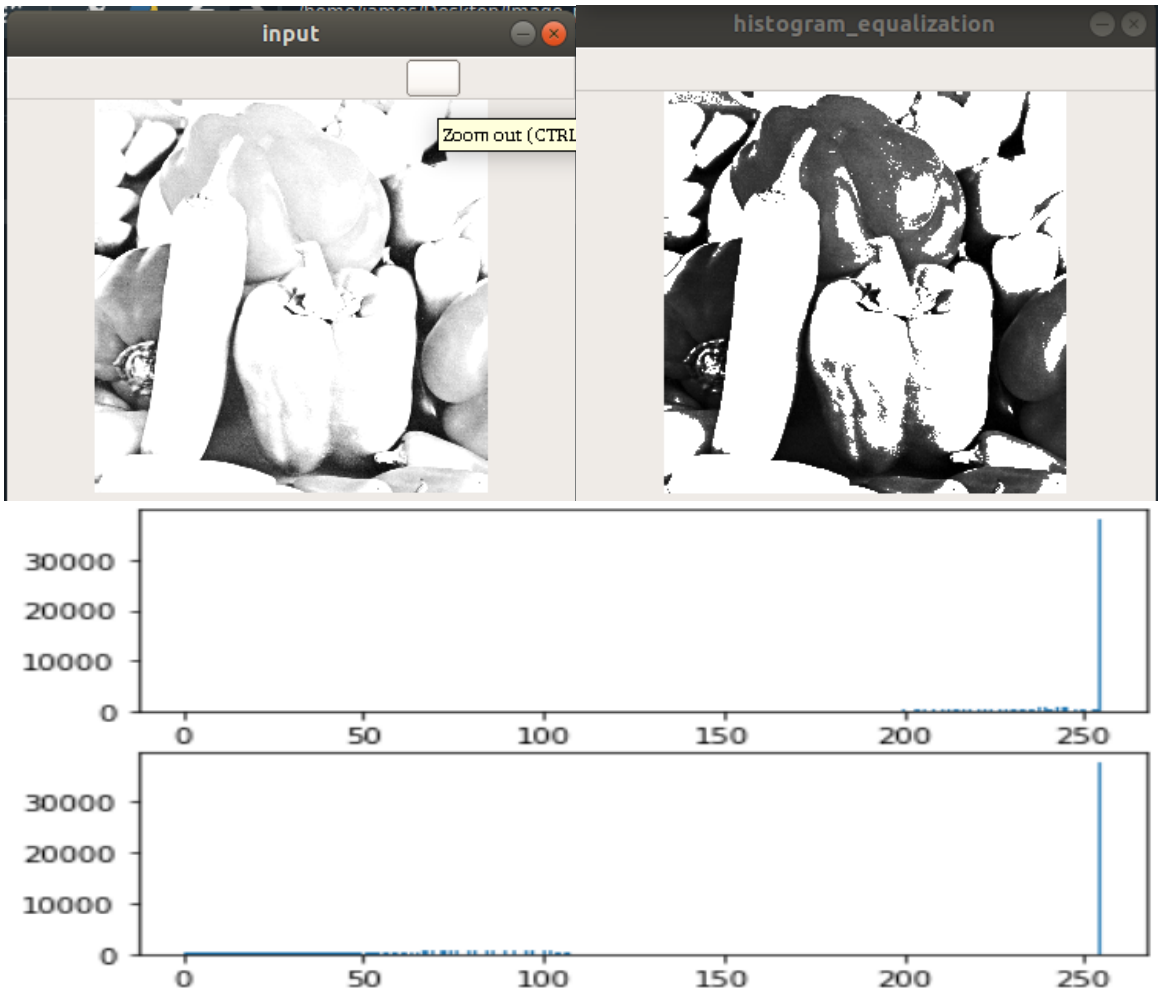


Cameraman.bmp y = 1.2

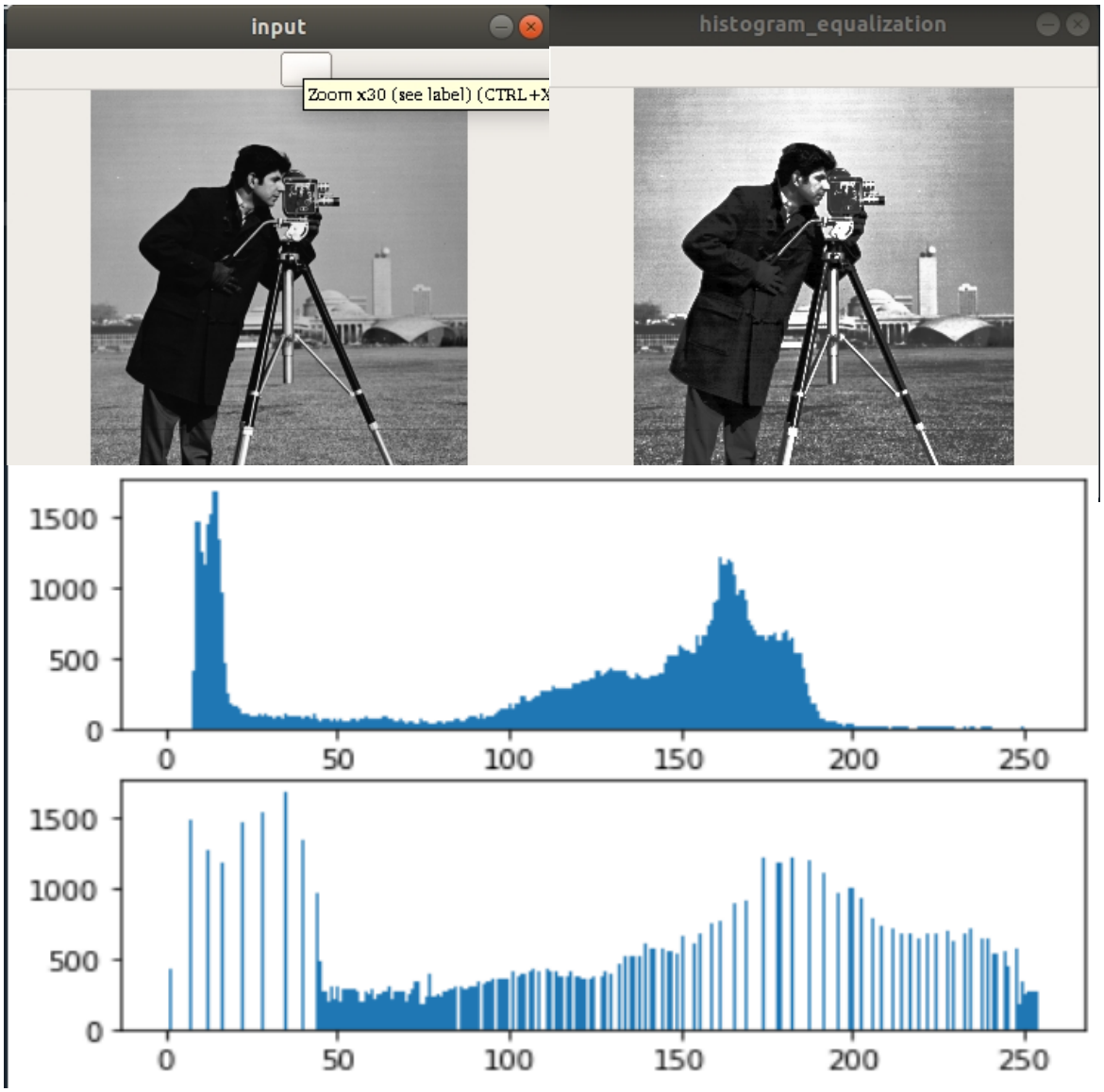
(2) histogram equalization

Lena.bmp

Peppers.bmp

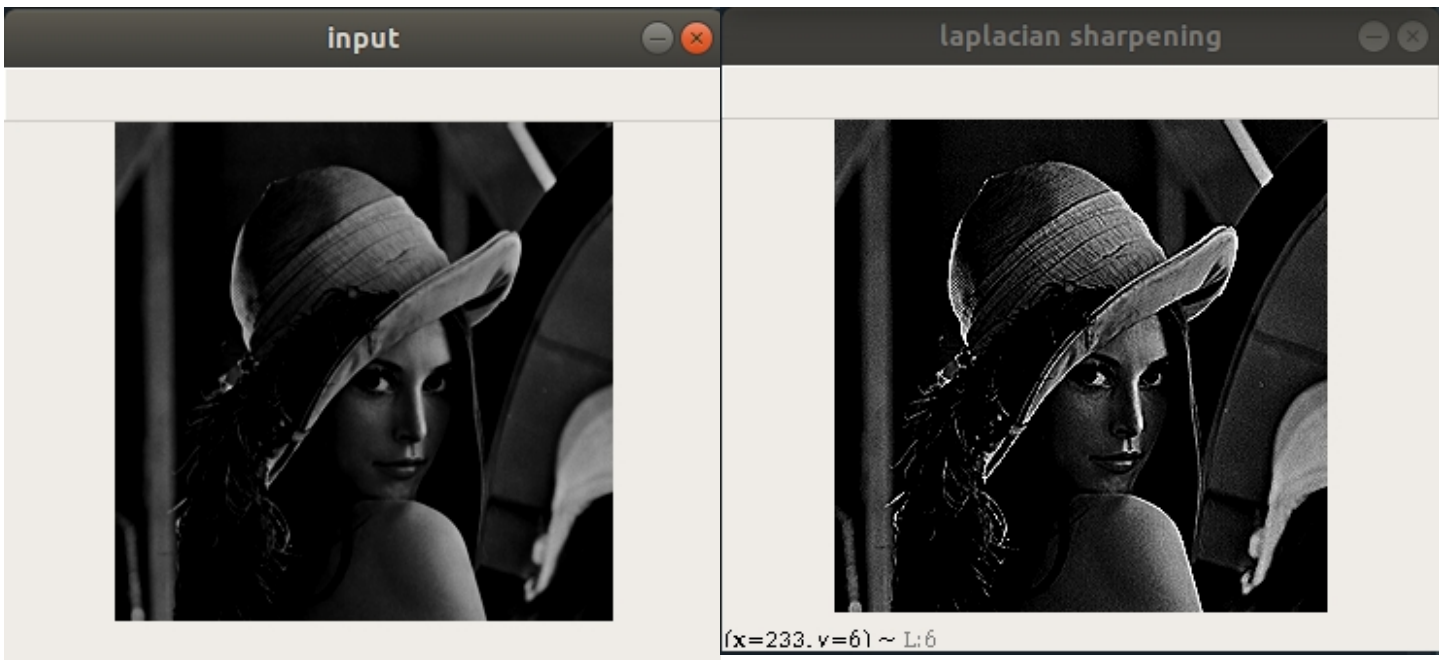


Cameraman.bmp

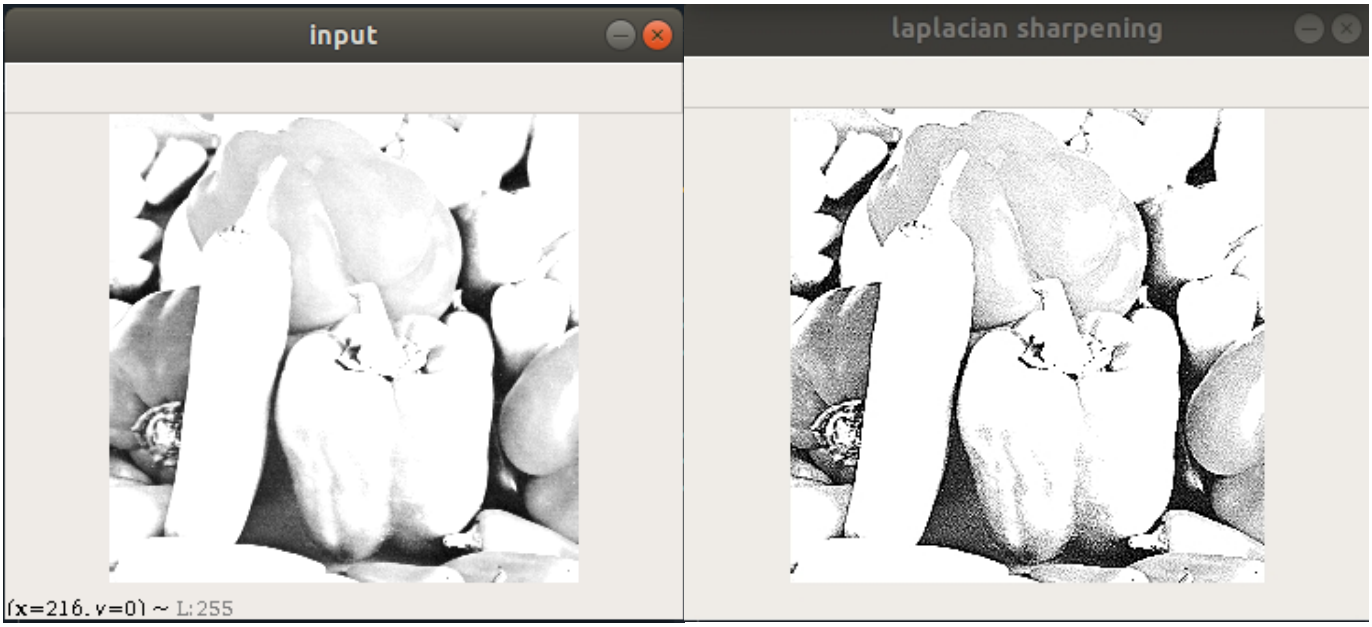


(3) image sharpening using the Laplacian

Lena.bmp



Peppers.bmp



Cameraman.bmp

Discussions:

針對三種圖像增強方法

(1) power-law (gamma) transformation

在正規化的指數轉換中，可以有效的增加對比度，但對過濾雜訊並不顯著。

(2) histogram equalization

直方圖均分中，以圖片本身的分佈機率去重新分配新像素，所以得以保留圖片的特徵。均勻化也因為會將集中的像素分散，有效增加對比度。

(3) image sharpening using the Laplacian

拉普拉斯的濾波器可以有效的將特徵擷取出來，但也會把雜訊強化出來，所以適合先進行模糊化的前處理將圖片提取特徵後與原圖相加就可以達到突顯特徵的效果。

References and Appendix:

https://blog.xuite.net/viplab/blog/307263602-Image+Enhancement+in+the+Spatial+Domain
http://iris123321.blogspot.com/2017/05/histogram-equalization.html
https://zh.wikipedia.org/wiki/%E7%9B%B4%E6%96%B9%E5%9B%BE%E5%9D%87%E8%A1%A1%E5%8C%96
https://ithelp.ithome.com.tw/articles/10192114
https://jason-chen-1992.weebly.com/home/-unsharp-masking
https://blog.csdn.net/chezhai/article/details/57514746