影像處理 Assignment2 - 影像銳化

姓名：吳嘉偉　學號：5105056013　日期：2017/12/31

1　灰階處理

1.1　把原始彩色影像轉成灰階影像

為了方便處理，所以先把彩色的影像轉成灰階影像

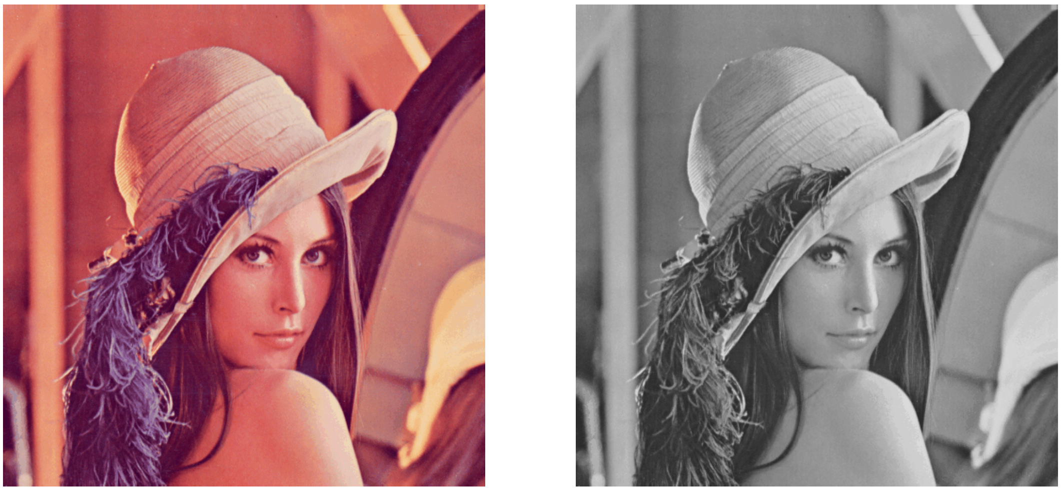

Figure 1: Source and Gray Image

1.2 程式碼

```
# 取得並儲存灰階照片
def readGrayImage(name):
    grayImage = cv2.imread(name, cv2.IMREAD_GRAYSCALE)
    savePhoto('gray_image', grayImage)
    return grayImage
```

2 Lapalcian Enhancement

2.1 Lapalcian Mask

用 Lapalcian Mask 二階微分找影像的邊緣

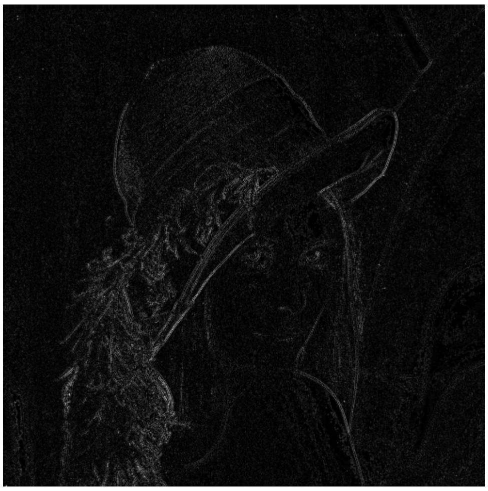

Figure 2: Lapalcian Mask Edge Image

程式碼

```
# Lapalcian Mask
def laplacianMask(image):
    lap = cv2.Laplacian(image, cv2.CV_64F)
    lap = np.uint8(np.absolute(lap))
    savePhoto('laplacian_mask_image', lap)
    return lap
```

2.2 實現 Lapalcian Enhancement

把轉成灰階的原始照片與用 Lapalcian Mask 產生的照片相加輸出的照片會有雜訊產生

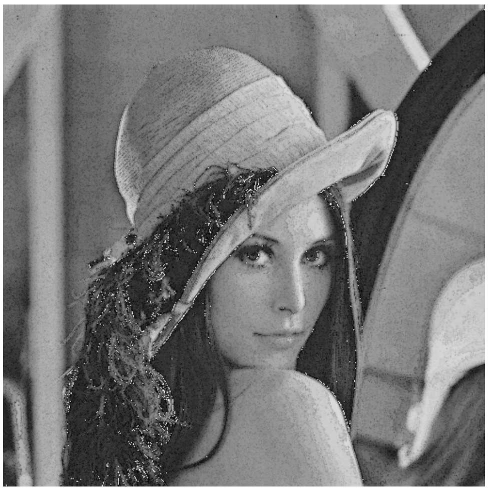

Figure 3: Lapalcian Enhancement Image

程式碼

```
def lap_enhance(source, lapacian):
    height = source.shape[0]
    width = source.shape[1]
    # height, width, _ = source.shape
    newImage = np.zeros((height, width, 3), np.uint8)
    for x in range(width):
        for y in range(height):
            value = source[y][x] - lapacian[y][x]
            if value > 255:
                value = 255
            else:
                value = int(value)
            newImage[y][x] = value
    savePhoto('lap_enhance_image', newImage)
    return newImage
```

3 Sobel Filter Enhancement

3.1 Sobel Filter

利用 Sobel Filter 找邊緣

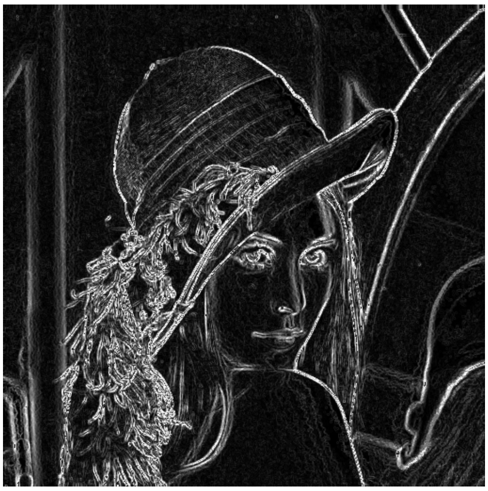

Figure 4: Sobel Filter Image

程式碼

```
# Sobel Filter
def sobelFilter(image):
    sobelX = cv2.Sobel(image, cv2.CV_64F, 1, 0)
    sobelY = cv2.Sobel(image, cv2.CV_64F, 0, 1)
    sobelX = np.uint8(np.absolute(sobelX))
    sobelY = np.uint8(np.absolute(sobelY))
    sobelCombined = cv2.bitwise_or(sobelX, sobelY)
    savePhoto('sobel_filter_image', sobelCombined)
```

3.2 Convolution

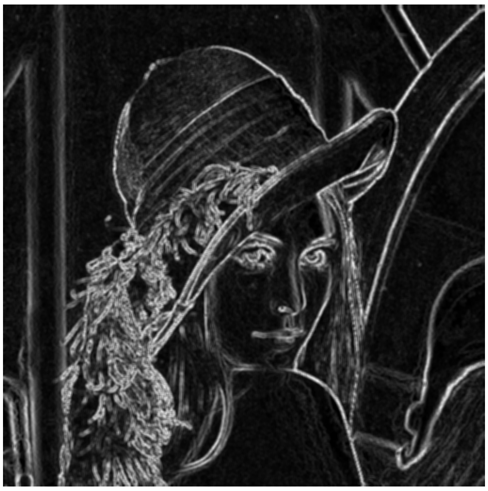

Figure 5: Convolution Image

程式碼

```
def convolution():
    image = cv2.imread('sobel_filter_image.png')
    blurred = cv2.blur(image, (3, 3))
    savePhoto('convolution_image', blurred)
    return blurred
```

3.3 Normalization

把影像做正規化，把 Convolution 模糊化後的影像乘以 Lapalcian Mask 的影像

程式碼

```
def normalization(source, blur):
    height = source.shape[0]
    width = source.shape[1]
    # height, width, _ = source.shape
    newImage = np.zeros((height, width, 3), np.uint8)
    for x in range(width):
        for y in range(height):
            newImage[y][x] = blur[y][x] / 255 * source[y][x]
    savePhoto('normalization_image', newImage)
    return newImage
```

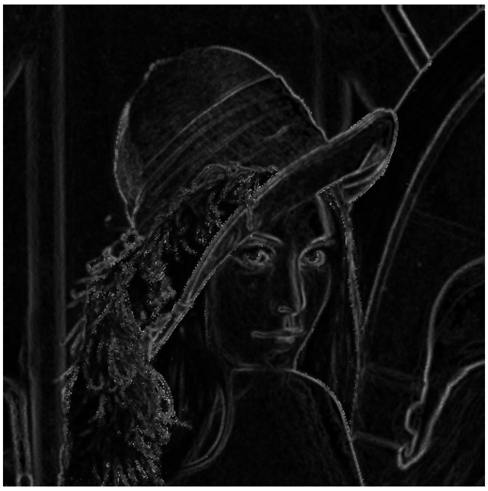

Figure 6: Normalization Image

3.4 Output

最後把正規化後的影像與原始照片相加，即可產生銳化的影像

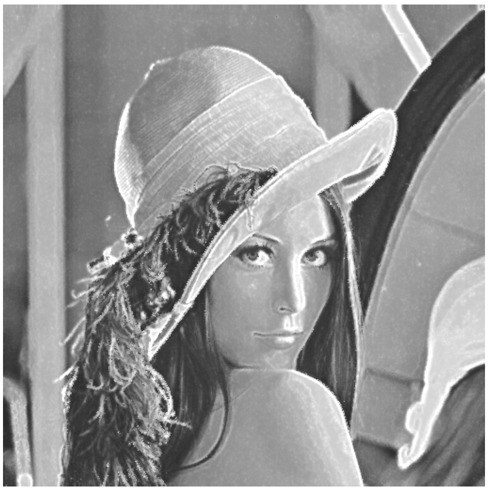

Figure 7: Final EnhancementImage Image

程式碼

```
def enhancementImage_New(source, normalization):
    height = source.shape[0]
    width = source.shape[1]
    # height, width, _ = source.shape
    newImage = np.zeros((height, width, 3), np.uint8)
    for x in range(width):
        for y in range(height):
```

```
tmp2 = int(normalization[y][x][0])
tmp1 = int(source[y][x])
tmp0 = tmp1 + tmp2
print(y, x)
if tmp0 > 255:
    newImage[y][x] = [255, 255, 255]
else:
    newImage[y][x] = normalization[y][x] + source[y][x]

savePhoto('FinalEnhancementImage', newImage)
```

4 結論

使用 Laplacian Enhancement 還是會有雜訊。
使用 Sobel Filter Enhancement 雜訊明顯比 Laplacian 少很多，而且還可以把影像銳化。
另外在 coding 時有發現兩張照片像素相加如果超過 255 時，python 會自動減掉 255，造成一開始輸出的影像都還是很多雜訊。

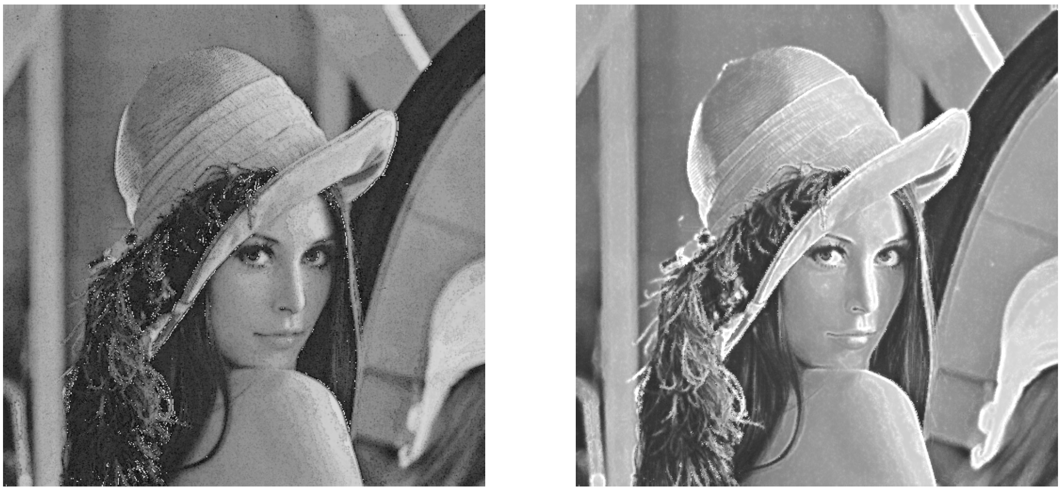

Figure 8: Laplacian and Sobel Enhancement Image