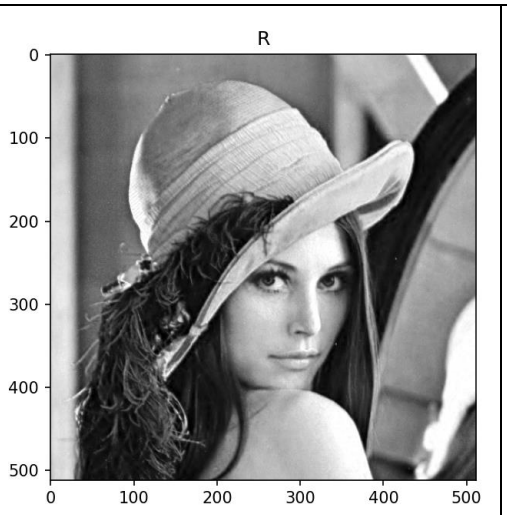
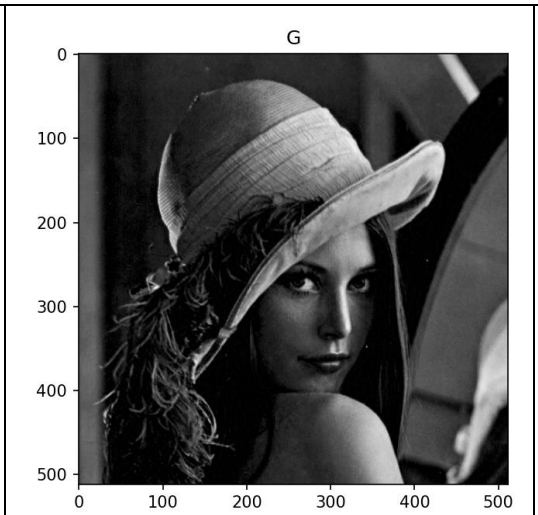
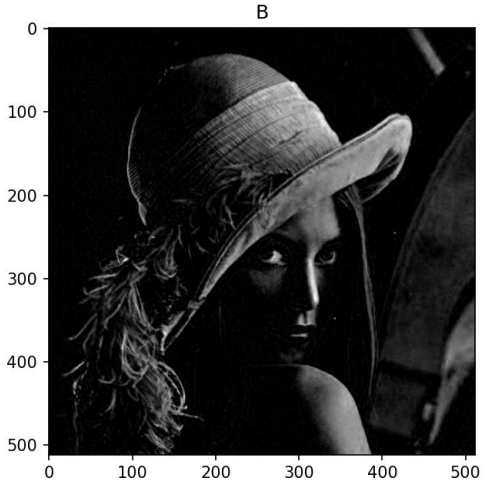
數位影像處理 DIP Homework Ch6

1. Please use a gradient computed in RGB color vector space to detect the edge for the image, 'lenna-RGB.tif' as Fig. 6.44(b) in pp.453. Please describe your method, procedures, final gradient image and print out the source code? (3X10=40)

原圖 lenna-RGB.tif







use a gradient computed in RGB color vector space to detect the edge for the image

方法: 參考 課本 p.451, p.452 計算 gradients(如下):

$$\mathbf{u} = \frac{\partial R}{\partial x}\mathbf{r} + \frac{\partial G}{\partial x}\mathbf{g} + \frac{\partial B}{\partial x}\mathbf{b} \tag{6-50}$$

and

$$\mathbf{v} = \frac{\partial R}{\partial y}\mathbf{r} + \frac{\partial G}{\partial y}\mathbf{g} + \frac{\partial B}{\partial y}\mathbf{b} \tag{6-51}$$

Let the quantities g_{xx}, g_{yy}, and g_{xy} be defined in terms of the dot product of these vectors, as follows:

$$g_{xx} = \mathbf{u}\cdot\mathbf{u} = \mathbf{u}^T\mathbf{u} = \left|\frac{\partial R}{\partial x}\right|^2 + \left|\frac{\partial G}{\partial x}\right|^2 + \left|\frac{\partial B}{\partial x}\right|^2 \tag{6-52}$$

$$g_{yy} = \mathbf{v}\cdot\mathbf{v} = \mathbf{v}^T\mathbf{v} = \left|\frac{\partial R}{\partial y}\right|^2 + \left|\frac{\partial G}{\partial y}\right|^2 + \left|\frac{\partial B}{\partial y}\right|^2 \tag{6-53}$$

and

$$g_{xy} = \mathbf{u}\cdot\mathbf{v} = \mathbf{u}^T\mathbf{v} = \frac{\partial R}{\partial x}\frac{\partial R}{\partial y} + \frac{\partial G}{\partial x}\frac{\partial G}{\partial y} + \frac{\partial B}{\partial x}\frac{\partial B}{\partial y} \tag{6-54}$$

by Di Zenzo [1986] for details. The Sobel operators discussed in Section 3.6 can be used to compute the partial derivatives required for implementing Eqs. (6-52) through (6-54).

Eqs (6-52~6-54) 在實作尚可用 Sobelx, Sobely 實現

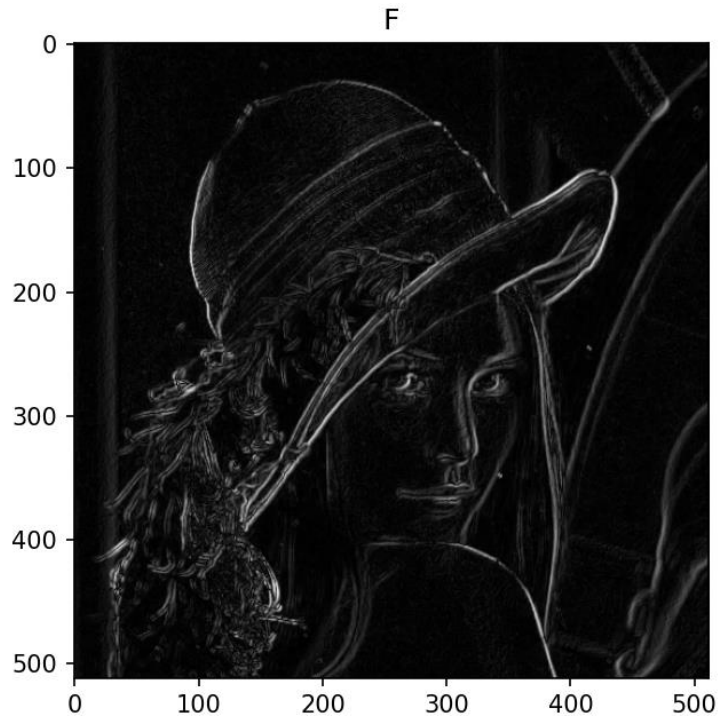
$$\theta(x,y) = \frac{1}{2}\tan^{-1}\left[\frac{2g_{xy}}{g_{xx} - g_{yy}}\right] \tag{6-55}$$

and that the value of the rate of change at (x, y) in the direction of $\theta(x, y)$ is given by

$$F_\theta(x,y) = \left\{\frac{1}{2}\left[\left(g_{xx} + g_{yy}\right) + \left(g_{xx} - g_{yy}\right)\cos 2\theta(x,y) + 2g_{xy}\sin 2\theta(x,y)\right]\right\}^{\frac{1}{2}} \tag{6-56}$$

pi?

最後所產生的 F(x, y)，即是用 **gradient computed in RGB color vector space** 所產生出的 edge detection 圖，結果如下頁。



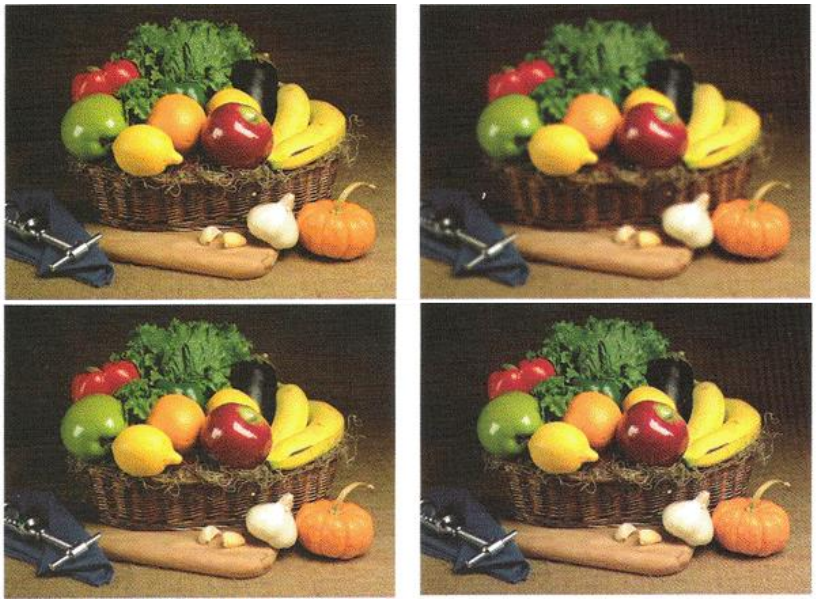

```python
import cv2
import numpy as np
import matplotlib.pyplot as plt

img = cv2.imread("lenna-RGB.tif")
img = img[:,:,::-1]   # BGR -> RGB

R = img[:,:,0]
G = img[:,:,1]
B = img[:,:,2]
```

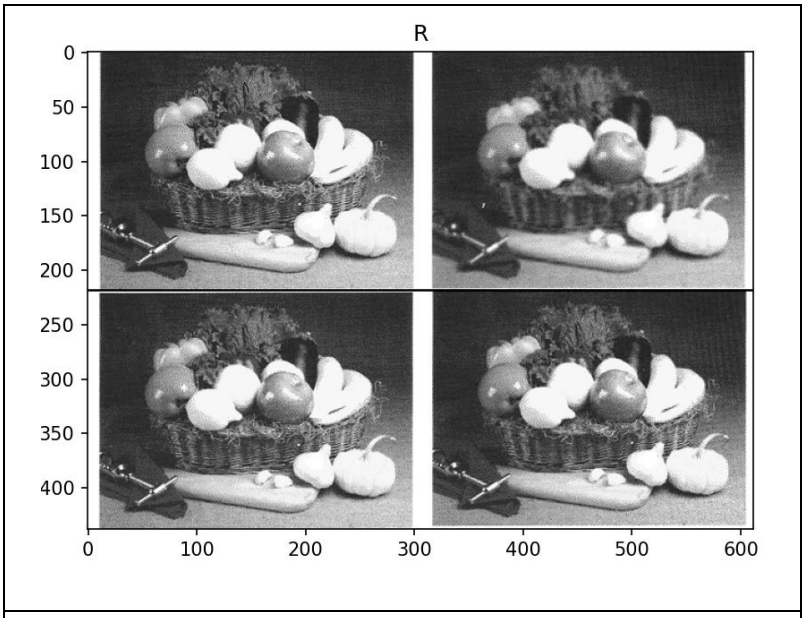
```python
# Sobely
r_Ry = cv2.Sobel(R, cv2.CV_32F, 0, 1, ksize=3)
g_Ry = cv2.Sobel(G, cv2.CV_32F, 0, 1, ksize=3)
b_Ry = cv2.Sobel(B, cv2.CV_32F, 0, 1, ksize=3)

# gxx, gyy, gxy
gxx = r_Rx**2 + g_Rx**2 + b_Rx**2   # same as gxx = abs(r_Rx)**2 + abs(g_Rx)**2 + abs(b_Rx)**2
gyy = r_Ry**2 + g_Ry**2 + b_Ry**2   # same as gyy = abs(r_Ry)**2 + abs(g_Ry)**2 + abs(b_Ry)**2
gxy = r_Rx*r_Ry + g_Rx*g_Ry + b_Rx*b_Ry

# angle(x, y)
angle = 1/2 * (np.arctan2((2*gxy),(gxx-gyy+1)))  # np.arctan2   # +1 prevent dividing zero
angle = abs(angle)
print(f"angle.max(), angle.min() = {angle.max(), angle.min()}")    # (1.5707964, 0.0)
print(f"angle.shape = {angle.shape}")                              # (512, 512)

# F(x, y)
F = (1/2* ( (gxx + gyy)+(gxx - gyy)*np.cos(2*angle) + 2*gxy*np.sin(2*angle) ) )**(1/2)
print(f"F.shape = {F.shape}")      # (512, 512)
plt.imshow(F , cmap="gray"), plt.title("F")
plt.show()
```

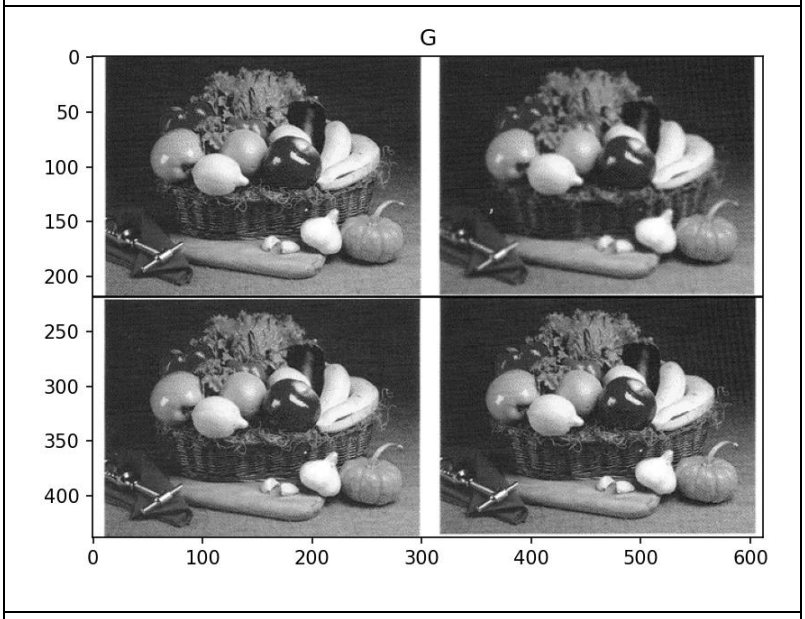
2. Repeat (1) steps in the image ‘Visual resolution.gif’? (3X10=40)

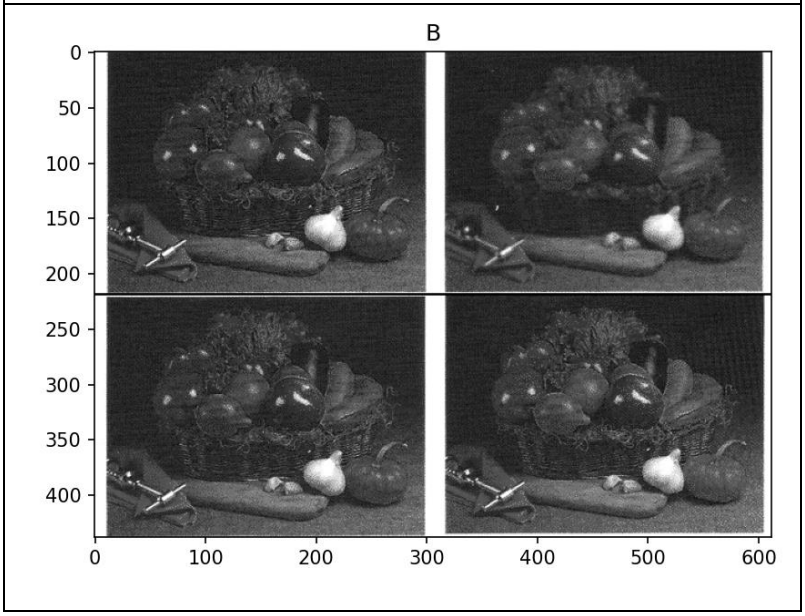
原圖 Visual resolution.gif

(這邊要特別注意原本拿到的圖的 shape 是(438, 612, 4), 第四個維度是一條水平的黑線，應該是誤植了，因此實際在做取前三個 channel 即可)







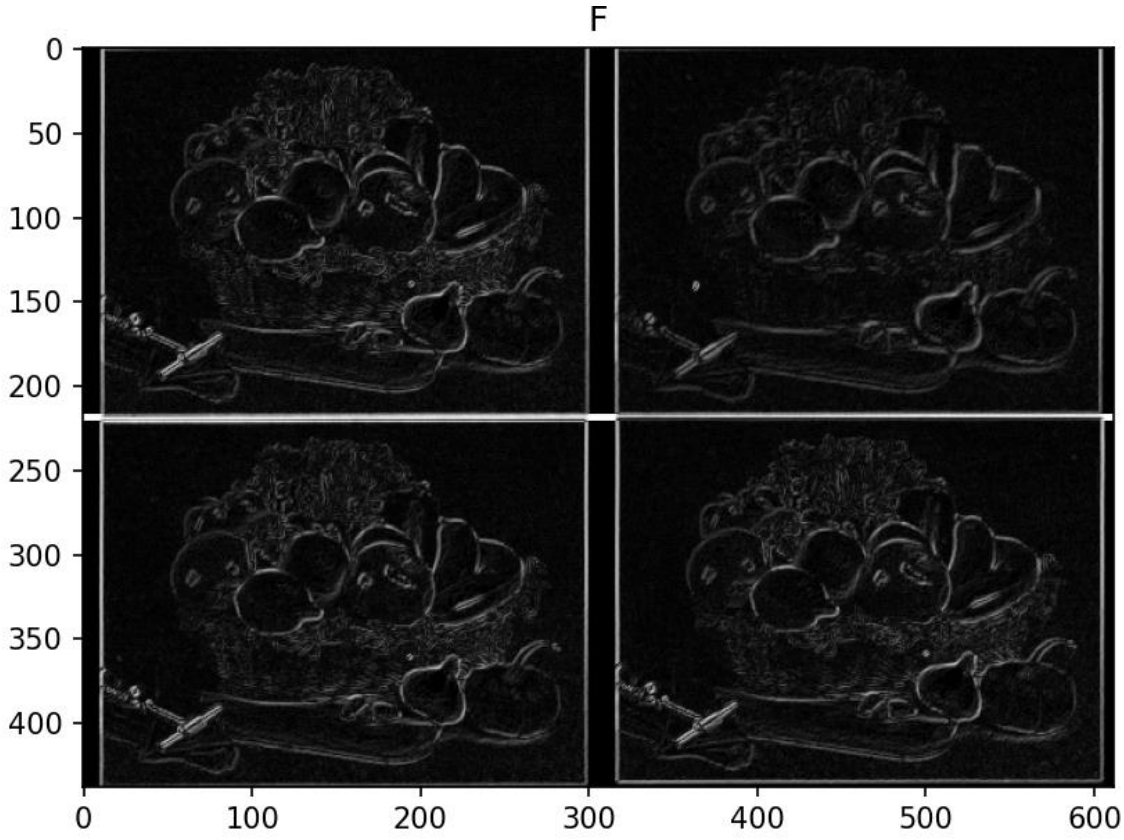


比較 R, G, B 三個 channel 的影像，可以發現 B channel 影像偏暗，這也合理，因為原圖幾乎沒有藍色(或由藍色混和成)的地方。

use a gradient computed in RGB color vector space to detect the edge for the image

方法: 同樣參考 課本 p.451, p.452 計算 gradients

公式及程式碼同上題



3. Please comment and compare your two designs? (20)

這邊想討論兩個問題:

1. np.arctan vs. np.arctan2
2. 比較使用 gradient computed in RGB color vector space 與 Sobel 做 edge detection 的差異

1. np.arctan vs. np.arctan2

在 python 套件裡有兩個可計算 arctan 的套件，兩者 return 的角度範圍不同:

np.arctan2 : The output of the function range from -180 to +180

np.arctan : The output of the function range from -90 to +90

主要是 np.arctan2 可 return 在四個象限中的角度，而 np.arctan 僅能 return 兩個象限中的角度

Reference :

https://geo.libretexts.org/Courses/University_of_California_Davis/GEL_056%3A_Introduction_to_Geophysics/Geophysics_is_everywhere_in_geology.../zz%3A_Back_Matter/Arctan_vs_Arctan2

而這邊我根據 課本 p.452 內容，**在實作上使用 np.arctan2，並取 abs**

Because $\tan(\alpha) = \tan(\alpha \pm \pi)$, if θ_0 is a solution to Eq. (6-55), so is $\theta_0 \pm \pi/2$. Furthermore, $F_\theta = F_{\theta+\pi}$, so F has to be computed only for values of θ in the half-open interval $[0, \pi)$. The fact that Eq. (6-55) gives two values 90° apart means that this

```
28    # angle(x, y)
29    angle = 1/2 * (np.arctan2((2*gxy),(gxx-gyy+1)))  # np.arctan2   # +1 prevent dividing zero
30    angle = abs(angle)
31    print(f"angle.max(), angle.min() = {angle.max(), angle.min()}")    # (1.5707964, 0.0)
```

2. 比較使用 gradient computed in RGB color vector space 與 Sobel 做 edge detection 的差異

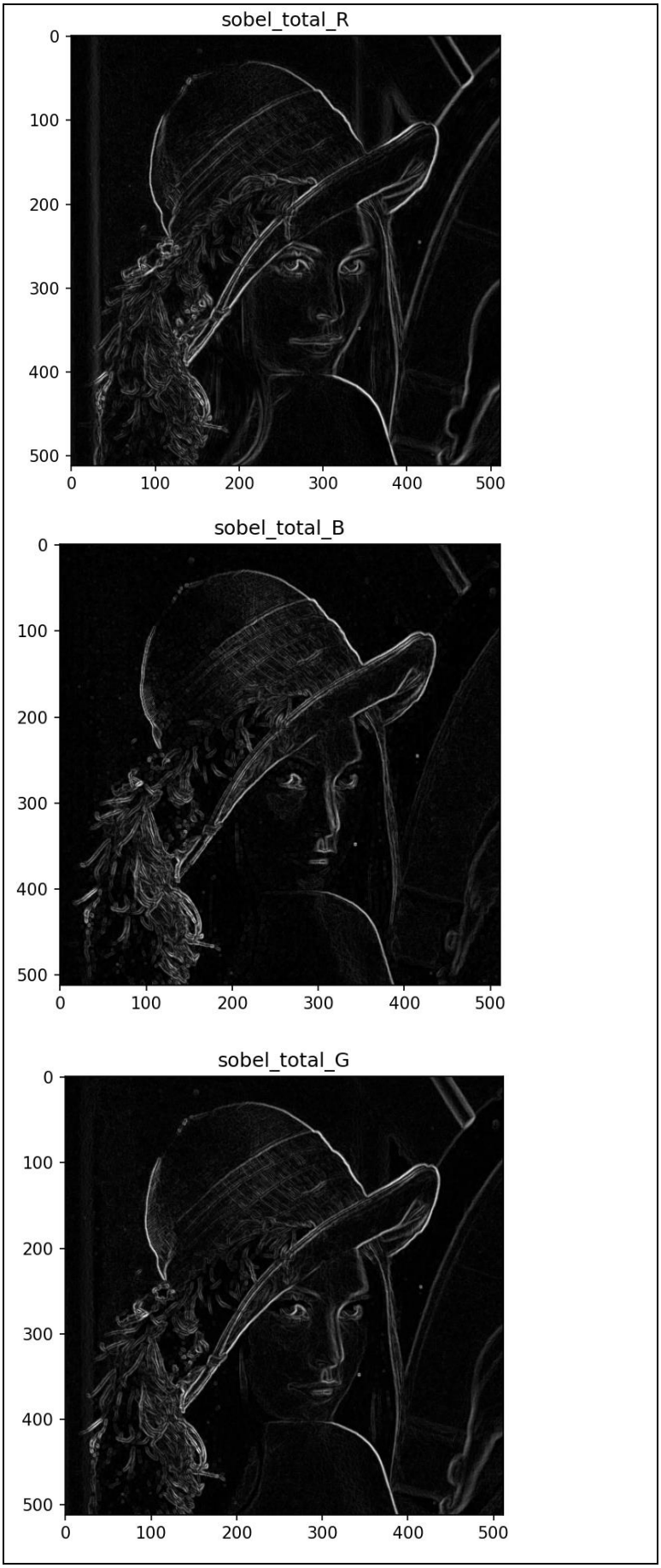
(以第一小題為例)

這邊試著做出課本 p.453 FIGURE 6.44(c) 的圖，依課本做法，分別在 R, G, B 3 channel 做 Sobel，再把三者相加起來。(Sobel 的實作與先前 HW3_4 的作法相同)

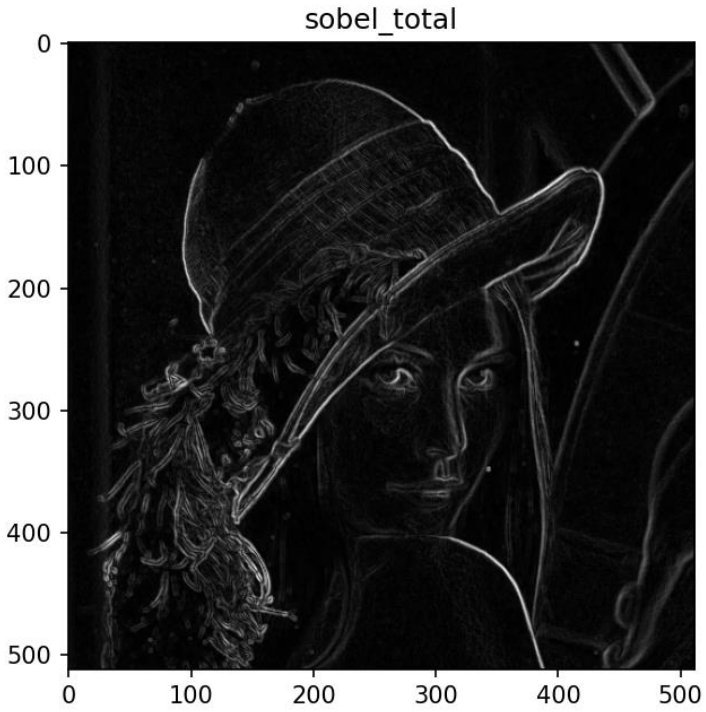
a b
c d

FIGURE 6.44
(a) RGB image. (b) Gradient computed in RGB color vector space. (c) Gradient image formed by the elementwise sum of three individual gradient images, each computed using the Sobel operators. (d) Difference between (b) and (c).

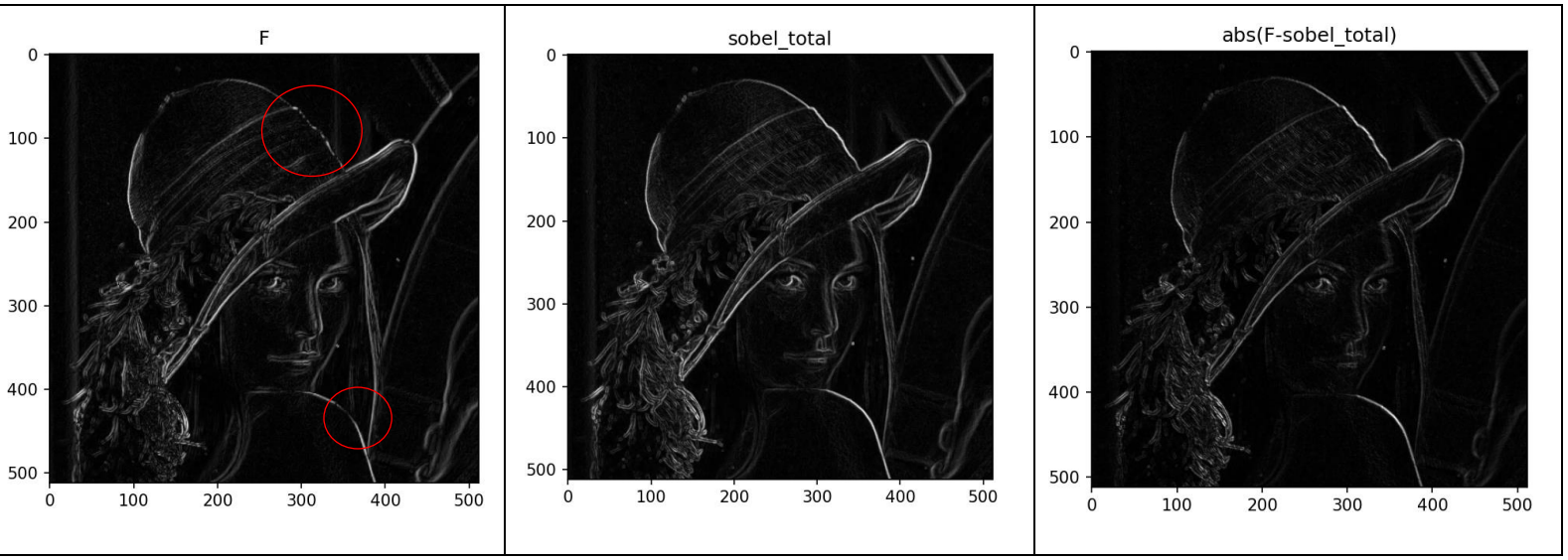
```
# ================ Sobel ================================================================================
## sobelx
sobelRx = cv2.Sobel(R, cv2.CV_32F, 1, 0, ksize=3)  # x
sobelGx = cv2.Sobel(G, cv2.CV_32F, 1, 0, ksize=3)
sobelBx = cv2.Sobel(B, cv2.CV_32F, 1, 0, ksize=3)

## sobely
sobelRy = cv2.Sobel(R, cv2.CV_32F, 0, 1, ksize=3)  # y
sobelGy = cv2.Sobel(G, cv2.CV_32F, 0, 1, ksize=3)
sobelBy = cv2.Sobel(B, cv2.CV_32F, 0, 1, ksize=3)

sobel_total_R = np.abs(sobelRx) + np.abs(sobelRy)
sobel_total_G = np.abs(sobelGx) + np.abs(sobelGy)
sobel_total_B = np.abs(sobelBx) + np.abs(sobelBy)

plt.imshow(sobel_total_R, cmap="gray"), plt.title("sobel_total_R")
plt.show()
plt.imshow(sobel_total_G, cmap="gray"), plt.title("sobel_total_G")
plt.show()
plt.imshow(sobel_total_B, cmap="gray"), plt.title("sobel_total_B")
plt.show()

## sobel_total
sobel_total = sobel_total_R + sobel_total_G + sobel_total_B
plt.imshow(sobel_total, cmap="gray"), plt.title("sobel_total")
plt.show()
```


sobel_total_R
0
100
200
300
400
500
sobel_total_B
sobel_total_G

用 Sobel 做出來的結果



比較使用 gradient computed in RGB color vector space 與 Sobel 做 edge detection 的差異



其實相較課本結果兩個方法做出來都比課本差，但 edge detection 的效果還是有出來，不過這邊做出來的結論與課本不同: Sobel 做出來的 edge detection 較細緻(也有將差異值 max, min print 出如下以證明)，這邊把一些差異用紅色圈出。

```
80    Difference = F-sobel_total
81    print(f"Difference.max(), Difference.min() = {Difference.max(), Difference.min()}")
```

```
Difference.max(), Difference.min() = (0.0, -2326.9004)
```