Computer Vision I (922 U0610) - Homework 9

Author: alanhc

ID: r10944007

Date: 10/27

README

1. create env: `conda env create -f environment.yml`
2. enter env: `conda activate ntu-cv`
3. run jupyter `jupyter notebook`

You are to implement following edge detectors with thresholds :

- (a) Robert's Operator: 12
- (b) Prewitt's Edge Detector: 24
- (c) Sobel's Edge Detector: 38
- (d) Frei and Chen's Gradient Operator: 30
- (e) Kirsch's Compass Operator: 135
- (f) Robinson's Compass Operator: 43
- (g) Nevatia-Babu 5x5 Operator: 12500

In [1]:
```
from PIL import Image
import numpy as np
import matplotlib.pyplot as plt
import cv2
import math


# Todo: 讀檔，確定影像大小
img = Image.open("input/lena.bmp")
img = np.array(img)

h, w = img.shape

print("image shape:", img.shape)
show = Image.fromarray(img).resize((256,256))
show
```

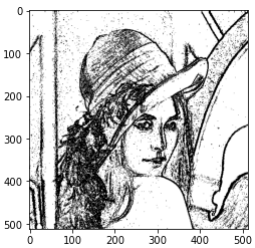
```
image shape: (512, 512)
```

Out[1]:

Robert's Operator: 12

In [2]:
```
img_padding = cv2.copyMakeBorder(img, 1, 1, 1, 1, cv2.BORDER_REPLICATE)
r_1 = np.array([[-1, 0], [ 0, 1]])
r_2 = np.array([[ 0,-1], [ 1, 0]])
h,w = img.shape

def window(img, y, x, kernel):
    now = 0
    for ky in range(kernel.shape[0]):
        for kx in range(kernel.shape[1]):
            now += img[y+ky][x+kx] * kernel[ky][kx]
    return now
ans = np.zeros((h,w))
for y in range(h):
    for x in range(w):
        # gradient magnitude
        now = math.sqrt(window(img_padding, y,x, r_1)**2+window(img_padding, y,x, r_2)**2)
        # 大於門檻設為0
        if now>=12:
            ans[y][x] = 0
        else:
            ans[y][x] = 255
plt.imshow(ans, cmap="gray")
Image.fromarray(ans).convert('RGB').save("output/Robert12.png")
```

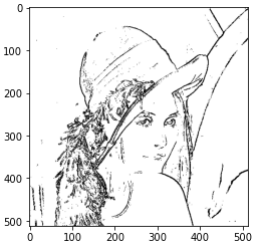


Robert's Operator: 30

In [3]:
```
img_padding = cv2.copyMakeBorder(img, 1, 1, 1, 1, cv2.BORDER_REPLICATE)
r_1 = np.array([[-1, 0], [ 0, 1]])
r_2 = np.array([[ 0,-1], [ 1, 0]])
h,w = img.shape

def window(img, y, x, kernel):
    now = 0
    for ky in range(kernel.shape[0]):
        for kx in range(kernel.shape[1]):
            now += img[y+ky][x+kx] * kernel[ky][kx]
    return now
ans = np.zeros((h,w))
for y in range(h):
    for x in range(w):
        # gradient magnitude
        now = math.sqrt(window(img_padding, y,x, r_1)**2+window(img_padding, y,x, r_2)**2)
        # 大於門檻設為0
        if now>=30:
```

```
            ans[y][x] = 0
        else:
            ans[y][x] = 255
plt.imshow(ans, cmap="gray")
Image.fromarray(ans).convert('RGB').save("output/Robert30.png")
```

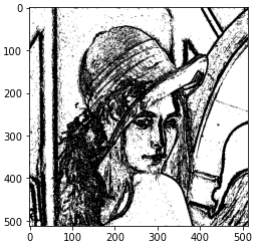


Prewitt's Edge Detector: 24

In [4]:

```
img_padding = cv2.copyMakeBorder(img, 1, 1, 1, 1, cv2.BORDER_REPLICATE)
p_1 = np.array([[-1, -1, -1], [ 0,  0,  0], [ 1,  1,  1]])
p_2 = np.array([[-1,  0,  1], [-1,  0,  1], [-1,  0,  1],])
h,w = img.shape

def window(img, y, x, kernel):
    now = 0
    for ky in range(kernel.shape[0]):
        for kx in range(kernel.shape[1]):
            now += img[y+ky][x+kx] * kernel[ky][kx]
    return now
ans = np.zeros((h,w))
for y in range(h):
    for x in range(w):
        # gradient magnitude
        now = math.sqrt(window(img_padding, y,x, p_1)**2+window(img_padding, y,x, p_2)**2)
        # 大於門檻設為0
        if now>=24:
            ans[y][x] = 0
        else:
            ans[y][x] = 255
plt.imshow(ans, cmap="gray")
Image.fromarray(ans).convert('RGB').save("output/Prewitt24.png")
```

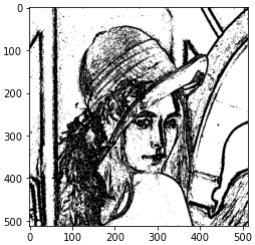


Sobel's Edge Detector: 38

In [5]:

```
img_padding = cv2.copyMakeBorder(img, 1, 1, 1, 1, cv2.BORDER_REPLICATE)
s_1 = np.array([[-1, -2, -1], [ 0,  0,  0], [ 1,  2,  1]])
s_2 = np.array([[-1,  0,  1], [-2,  0,  2], [-1,  0,  1],])
h,w = img.shape

def window(img, y, x, kernel):
    now = 0
    for ky in range(kernel.shape[0]):
        for kx in range(kernel.shape[1]):
            now += img[y+ky][x+kx] * kernel[ky][kx]
    return now
ans = np.zeros((h,w))
for y in range(h):
    for x in range(w):
        # gradient magnitude
        now = math.sqrt(window(img_padding, y,x, s_1)**2+window(img_padding, y,x, s_2)**2)
        # 大於門檻設為0
        if now>=38:
            ans[y][x] = 0
        else:
            ans[y][x] = 255
plt.imshow(ans, cmap="gray")
Image.fromarray(ans).convert('RGB').save("output/Sobel38.png")
```



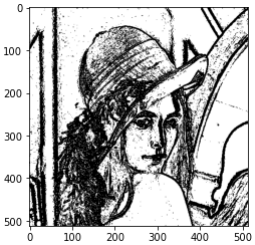
Frei and Chen's Gradient Operator: 30

In [6]:

```
img_padding = cv2.copyMakeBorder(img, 1, 1, 1, 1, cv2.BORDER_REPLICATE)
f_1 = np.array([[-1, -math.sqrt(2), -1], [ 0,  0,  0], [ 1,  math.sqrt(2),  1]])
f_2 = np.array([[-1,  0,  1], [-math.sqrt(2),  0, math.sqrt(2)], [-1,  0,  1],])
h,w = img.shape

def window(img, y, x, kernel):
    now = 0
    for ky in range(kernel.shape[0]):
        for kx in range(kernel.shape[1]):
            now += img[y+ky][x+kx] * kernel[ky][kx]
    return now
ans = np.zeros((h,w))
for y in range(h):
    for x in range(w):
        # gradient magnitude
        now = math.sqrt(window(img_padding, y,x, f_1)**2+window(img_padding, y,x, f_2)**2)
        # 大於門檻設為0
        if now>=30:
            ans[y][x] = 0
```

```
        else:
            ans[y][x] = 255
plt.imshow(ans, cmap="gray")
Image.fromarray(ans).convert('RGB').save("output/Frei_and_Chen30.png")
```

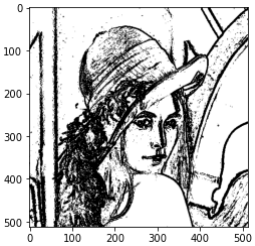


Kirsch's Compass Operator: 135

In [7]:

```
img_padding = cv2.copyMakeBorder(img, 1, 1, 1, 1, cv2.BORDER_REPLICATE)
k_0 = np.array([[-3,-3, 5],[-3, 0, 5],[-3,-3, 5]])
k_1 = np.array([[-3, 5, 5],[-3, 0, 5],[-3,-3,-3]])
k_2 = np.array([[ 5, 5, 5],[-3, 0,-3],[-3,-3,-3]])
k_3 = np.array([[ 5, 5,-3],[ 5, 0,-3],[-3,-3,-3]])
k_4 = np.array([[ 5,-3,-3],[ 5, 0,-3],[ 5,-3,-3]])
k_5 = np.array([[-3,-3,-3],[ 5, 0,-3],[ 5, 5,-3]])
k_6 = np.array([[-3,-3,-3],[-3, 0,-3],[ 5, 5, 5]])
k_7 = np.array([[-3,-3,-3],[-3, 0, 5],[-3, 5, 5]])
k_k = [k_0,k_1,k_2,k_3,k_4,k_5,k_6,k_7]
h,w = img.shape

def window(img, y, x, kernel):
    now = 0
    for ky in range(kernel.shape[0]):
        for kx in range(kernel.shape[1]):
            now += img[y+ky][x+kx] * kernel[ky][kx]
    return now
ans = np.zeros((h,w))
for y in range(h):
    for x in range(w):
        # 取大的
        _max = []
        for i in range(8):
            _max.append(window(img_padding, y,x, k_k[i]))
        now = max(_max)
        # 大於門檻設為0
        if now>=135:
            ans[y][x] = 0
        else:
            ans[y][x] = 255
plt.imshow(ans, cmap="gray")
Image.fromarray(ans).convert('RGB').save("output/Kirsch135.png")
```



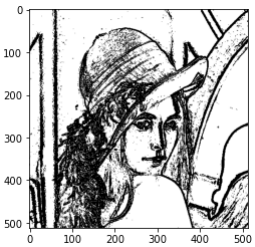
Robinson's Compass Operator: 43

In [8]:

```
img_padding = cv2.copyMakeBorder(img, 1, 1, 1, 1, cv2.BORDER_REPLICATE)
r_0 = np.array([[-1, 0, 1],[-2, 0, 2],[-1, 0, 1]])
r_1 = np.array([[ 0, 1, 2],[-1, 0, 1],[-2,-1, 0]])
r_2 = np.array([[ 1, 2, 1],[ 0, 0, 0],[-1,-2,-1]])
r_3 = np.array([[ 2, 1, 0],[ 1, 0,-1],[ 0,-1,-2]])
r_4 = - r_0
r_5 = - r_1
r_6 = - r_2
r_7 = - r_3

r_k = [r_0,r_1,r_2,r_3,r_4,r_5,r_6,r_7]
h,w = img.shape

def window(img, y, x, kernel):
    now = 0
    for ky in range(kernel.shape[0]):
        for kx in range(kernel.shape[1]):
            now += img[y+ky][x+kx] * kernel[ky][kx]
    return now
ans = np.zeros((h,w))
for y in range(h):
    for x in range(w):
        # 取大的
        _max = []
        for i in range(8):
            _max.append(window(img_padding, y,x, r_k[i]))
        now = max(_max)
        # 大於門檻設為0
        if now>=43:
            ans[y][x] = 0
        else:
            ans[y][x] = 255
plt.imshow(ans, cmap="gray")
Image.fromarray(ans).convert('RGB').save("output/Robinson43.png")
```



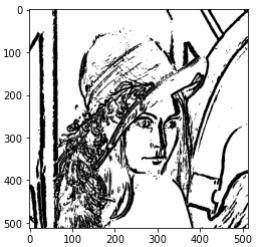
Nevatia-Babu 5x5 Operator: 12500

In [9]:

```
img_padding = cv2.copyMakeBorder(img, 2, 2, 2, 2, cv2.BORDER_REPLICATE)
n_0 = np.array([[ 100, 100, 100, 100, 100],[ 100, 100, 100, 100, 100],[   0,   0,   0,   0,   0],[-100,-100,-100,-100,-100],[-100,-100,-100,-100,-100]])
n_1 = np.array([[ 100, 100, 100, 100, 100],[ 100, 100, 100,  78, -32],[ 100,  92,   0, -92,-100],[  32, -78,-100,-100,-100],[-100,-100,-100,-100,-100]])
n_2 = np.array([[ 100, 100, 100,  32,-100],[ 100, 100,  92, -78,-100],[ 100, 100,   0,-100,-100],[ 100,  78, -92,-100,-100],[ 100, -32,-100,-100,-100]])
n_3 = np.array([[-100,-100,   0, 100, 100],[-100,-100,   0, 100, 100],[-100,-100,   0, 100, 100],[-100,-100,   0, 100, 100],[-100,-100,   0, 100, 100]])
n_4 = np.array([[-100,  32, 100, 100, 100],[-100, -78,  92, 100, 100],[-100,-100,   0, 100, 100],[-100,-100, -92,  78, 100],[-100,-100,-100, -32, 100]])
n_5 = np.array([[ 100, 100, 100, 100, 100],[ -32,  78, 100, 100, 100],[-100,-92,    0,  92, 100],[-100,-100,-100, -78,  32],[-100,-100,-100,-100,-100]])

n_k = [n_0,n_1,n_2,n_3,n_4,n_5]
h,w = img.shape

def window(img, y, x, kernel):
    now = 0
    for ky in range(kernel.shape[0]):
        for kx in range(kernel.shape[1]):
            now += img[y+ky][x+kx] * kernel[ky][kx]
    return now
ans = np.zeros((h,w))
for y in range(h):
    for x in range(w):
        # 取大的
        _max = []
        for i in range(6):
            _max.append(window(img_padding, y,x, n_k[i]))
        now = max(_max)
        # 大於門檻設為0
        if now>=12500:
            ans[y][x] = 0
        else:
            ans[y][x] = 255
plt.imshow(ans, cmap="gray")
Image.fromarray(ans).convert('RGB').save("output/Nevatia_Babu12500.png")
```



Ref

- textbook
- ppt