HW9

- **A discription of your homework**

  Programming language used: Python 2.7

  Library used: Numpy, PIL, Scipy.misc, math

- **Your parameters**

  i: row

  j: column

  tem: 用於儲存每個像素的灰階數值 0~255

  mask: masks for calculating gradients

  gra: gradient

  gra_max: the maximum of gradients calculated by different masks

  threshold: threshold for each kind of operator

- **Functions**

  roberts: Roberts operator

  prewitt: Prewitt edge detector

  sobel: Sobel edge detector

  frei_and_chen: Frei and Chen gradient operator

  kirsch: Kirsch compass operator

  robinson: Robinson compass operator

  nevatia_babu: Nevatia-Babu 5X5 operator

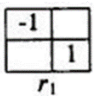
- **The algorithm you used**

  Apply the masks to each pixel of lena: multiply the elements of masks to the gray scales of lena and sum up the values, so we get $\mathrm{r}_1, \mathrm{r}_2$ in Roberts operator, for instance.

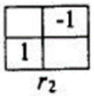
  If gradient > threshold, make that pixel black. Otherwise, make that pixel white.

(1) Roberts operator

mask: (i,j) corresponds to the left-up corner of the mask



r_1



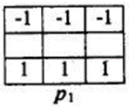
r_2

$$\text{gradient} = \sqrt{r_1^2 + r_2^2}$$

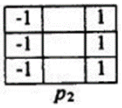
threshold=30

(2) Prewitt edge detector

mask: (i,j) corresponds to the center of the mask



p_1



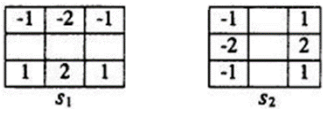
p_2

$$\text{gradient} = \sqrt{p_1^2 + p_2^2}$$

threshold=90

(3) Sobel edge detector

mask: (i,j) corresponds to the center of the mask

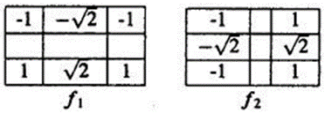


$$\text{gradient} = \sqrt{s_1^2 + s_2^2}$$

threshold=120

(4) Frei and Chen gradient operator

mask: (i,j) corresponds to the center of the mask

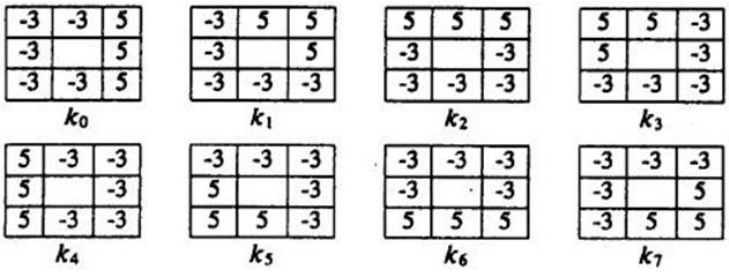


$$\text{gradient} = \sqrt{f_1^2 + f_2^2}$$

threshold=100

(5) Kirsch compass operator

mask: (i,j) corresponds to the center of the mask



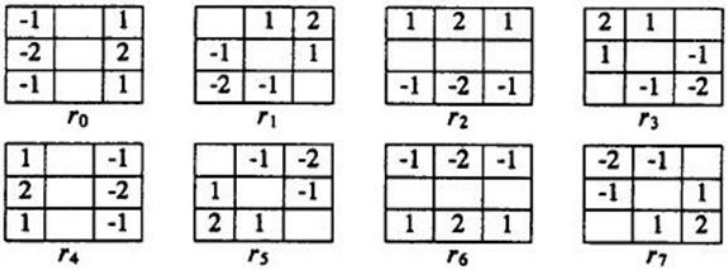
$$\text{gradient} = \max_{n=0,\ldots,7} k_n$$

If the gradient of k_n is larger than the one before, then save it to gra_mask and compared gra_mask to threshold.

threshold=430

(6) Robinson compass operator

mask: (i,j) corresponds to the center of the mask



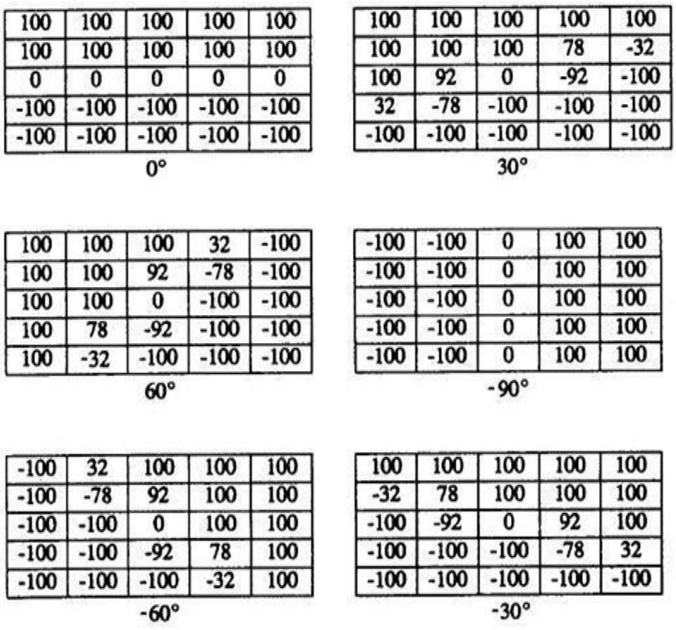
$$\text{gradient} = \max_{n=0,\ldots,7} k_n$$

If the gradient of k_n is larger than the one before, then save it to gra_mask and compared gra_mask to threshold.

threshold=120

(7) Nevatia-Babu 5X5 operator

mask: (i,j) corresponds to the center of the mask



the other 6 masks are the derived by multiplying -1 to the above 6 masks

$$\text{gradient} = \max_{n=0,\ldots,11} k_n$$

If the gradient of k_n is larger than the one before, then save it to gra_mask and compared gra_mask to threshold.

threshold=37000

- **Principal code fragment**

```
def roberts(x):
    tem=np.ones(x.shape)
    threshold = 30 #12
    mask_1=[[-1,0],[0,1]]
    mask_2=[[0,-1],[1,0]]
    for i in range(x.shape[0]-1):
        for j in range(x.shape[1]-1):
            y=x[i:i+2,j:j+2]
            gra = sqrt(np.sum(np.multiply(y,mask_1))**2+np.sum(np.multiply(y,mask_2))**2)
            if gra>threshold:
                tem[i][j]=0
    return tem

def prewitt(x):
    tem=np.ones(x.shape)
    threshold = 90 #24
    mask_1=[[-1,-1,-1],[0,0,0],[1,1,1]]
    mask_2=[[-1,0,1],[-1,0,1],[-1,0,1]]
    for i in range(1,x.shape[0]-1):
        for j in range(1,x.shape[1]-1):
            y=x[i-1:i+2,j-1:j+2]
            gra = sqrt(np.sum(np.multiply(y,mask_1))**2+np.sum(np.multiply(y,mask_2))**2)
            if gra>threshold:
                tem[i][j]=0
    return tem

def sobel(x):
    tem=np.ones(x.shape)
    threshold = 120 #38
    mask_1=[[-1,-2,-1],[0,0,0],[1,2,1]]
    mask_2=[[-1,0,1],[-2,0,2],[-1,0,1]]
    for i in range(1,x.shape[0]-1):
        for j in range(1,x.shape[1]-1):
            y=x[i-1:i+2,j-1:j+2]
            gra = sqrt(np.sum(np.multiply(y,mask_1))**2+np.sum(np.multiply(y,mask_2))**2)
            if gra>threshold:
                tem[i][j]=0
    return tem


def frei_and_chen(x):
    tem=np.ones(x.shape)
    threshold = 100 #30
    mask_1=[[-1,-sqrt(2),-1],[0,0,0],[1,sqrt(2),1]]
    mask_2=[[-1,0,1],[-sqrt(2),0,sqrt(2)],[-1,0,1]]
    for i in range(1,x.shape[0]-1):
        for j in range(1,x.shape[1]-1):
            y=x[i-1:i+2,j-1:j+2]
            gra = sqrt(np.sum(np.multiply(y,mask_1))**2+np.sum(np.multiply(y,mask_2))**2)
            if gra>threshold:
                tem[i][j]=0
    return tem

def kirsch(x):
    tem=np.ones(x.shape)
    threshold = 430 #135
    mask=[]#list
    mask.append(np.array([[-3,-3,5],[-3,0,5],[-3,-3,5]]))#k0
    mask.append(np.array([[-3,5,5],[-3,0,5],[-3,-3,-3]]))#k1
    mask.append(np.array([[5,5,5],[-3,0,-3],[-3,-3,-3]]))#k2
    mask.append(np.array([[5,5,-3],[5,0,-3],[-3,-3,-3]]))#k3
    mask.append(np.array([[5,-3,-3],[5,0,-3],[5,-3,-3]]))#k4
    mask.append(np.array([[-3,-3,-3],[5,0,-3],[5,5,-3]]))#k5
    mask.append(np.array([[-3,-3,-3],[-3,0,-3],[5,5,5]]))#k6
    mask.append(np.array([[-3,-3,-3],[-3,0,5],[-3,5,5]]))#k7
    for i in range(1,x.shape[0]-1):
        for j in range(1,x.shape[1]-1):
            gra_max=0
            for k in range(len(mask)):
                y=x[i-1:i+2,j-1:j+2]
                gra=np.sum(np.multiply(y,mask[k]))
                if gra>gra_max:
                    gra_max=gra
            if gra_max>threshold:
                tem[i][j]=0
    return tem
```

```
def robinson(x):
    tem=np.ones(x.shape)
    threshold = 120 #43
    mask=[]#list
    mask.append(np.array([[-1,0,1],[-2,0,2],[-1,0,1]]))#k0
    mask.append(np.array([[0,1,2],[-1,0,1],[-2,-1,0]]))#k1
    mask.append(np.array([[1,2,1],[0,0,0],[-1,-2,-1]]))#k2
    mask.append(np.array([[2,1,0],[1,0,-1],[0,-1,-2]]))#k3
    for k in range(len(mask)):
        mask.append(-mask[k])
    for i in range(1,x.shape[0]-1):
        for j in range(1,x.shape[1]-1):
            gra_max=0
            for k in range(len(mask)):
                y=x[i-1:i+2,j-1:j+2]
                gra = np.sum(np.multiply(y,mask[k]))
                if gra>gra_max:
                    gra_max=gra
            if gra_max>threshold:
                tem[i][j]=0
    return tem

def nevatia_babu(x):
    tem=np.ones(x.shape)
    threshold = 37000 #12500
    mask=[]#list
    mask.append(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]]))#k0
    mask.append(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]]))#k1
    mask.append(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]]))#k2
    mask.append(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]]))#k3
    mask.append(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]]))#k4
    mask.append(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]]))#k5
    for k in range(len(mask)):
        mask.append(-mask[k])
    for i in range(2,x.shape[0]-2):
        for j in range(2,x.shape[1]-2):
            gra_max=0
            for k in range(len(mask)):
                y=x[i-2:i+3,j-2:j+3]
                gra = np.sum(np.multiply(y,mask[k]))
                if gra>gra_max:
                    gra_max=gra
            if gra_max>threshold:
                tem[i][j]=0
    return tem
```

- **Resulting images**

1. Roberts operator

2. Prewitt edge detector

3. Sobel edge detector

4. Frei and Chen gradient operator

5. Kirsch compass operator

6. Robinson compass operator

7. Nevatia-Babu 5X5 operator