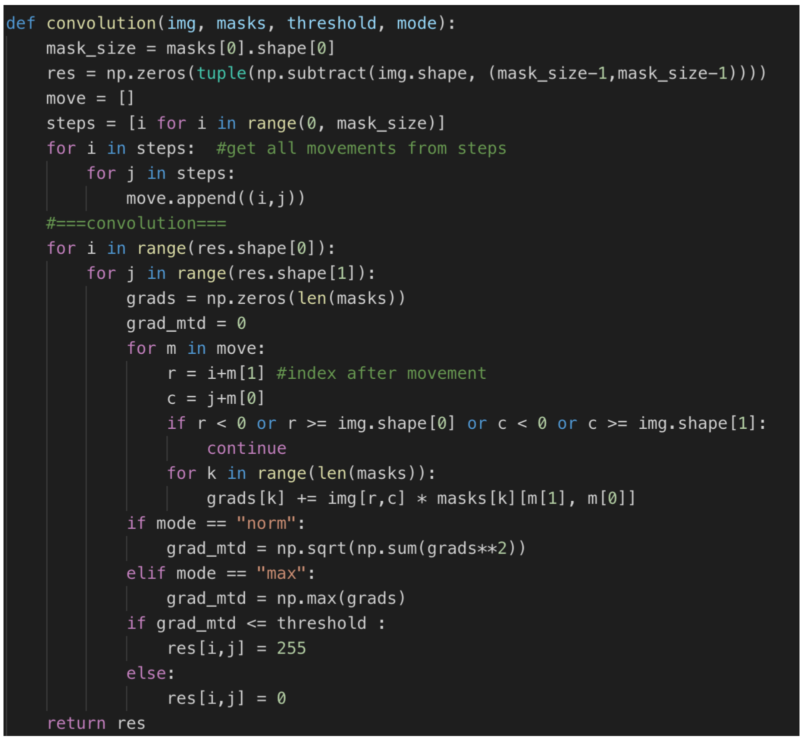
Computer and Robot Vision Homework 9
General Edge Detection
資工碩一 張家源 R07922102

使用語言： **Python**

Robert's Operator

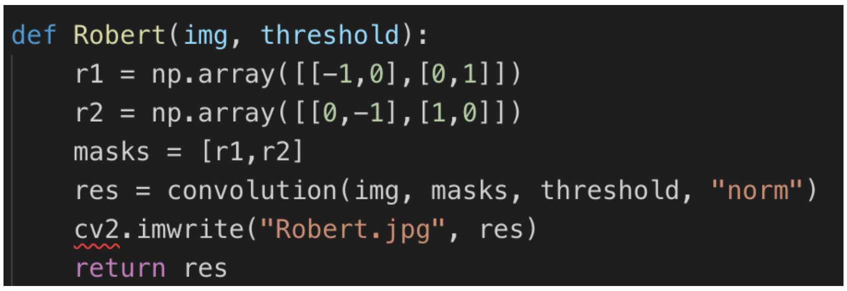
Threshold = 30

先實作Convolution的Function，供其他operator使用
- 根據所給的masks，對原圖用不同的mask作convolution運算得到每個mask之gradient
- 根據不同的operator，計算對應的gradient magnitude
- 檢查gradient magnitude 是否超過threshold

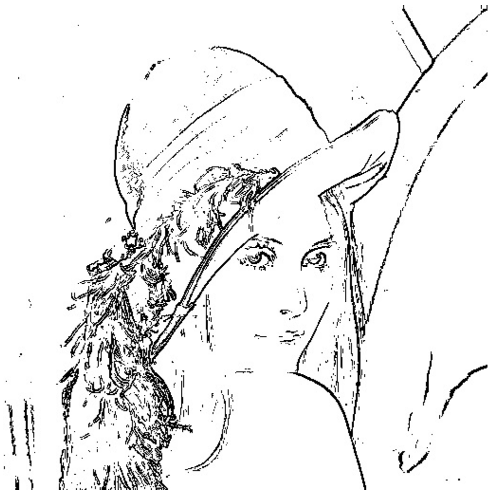


```python
def convolution(img, masks, threshold, mode):
    mask_size = masks[0].shape[0]
    res = np.zeros(tuple(np.subtract(img.shape, (mask_size-1,mask_size-1))))
    move = []
    steps = [i for i in range(0, mask_size)]
    for i in steps:  #get all movements from steps
        for j in steps:
            move.append((i,j))
    #===convolution===
    for i in range(res.shape[0]):
        for j in range(res.shape[1]):
            grads = np.zeros(len(masks))
            grad_mtd = 0
            for m in move:
                r = i+m[1] #index after movement
                c = j+m[0]
                if r < 0 or r >= img.shape[0] or c < 0 or c >= img.shape[1]:
                    continue
                for k in range(len(masks)):
                    grads[k] += img[r,c] * masks[k][m[1], m[0]]
            if mode == "norm":
                grad_mtd = np.sqrt(np.sum(grads**2))
            elif mode == "max":
                grad_mtd = np.max(grads)
            if grad_mtd <= threshold :
                res[i,j] = 255
            else:
                res[i,j] = 0
    return res
```

Robert Function Code

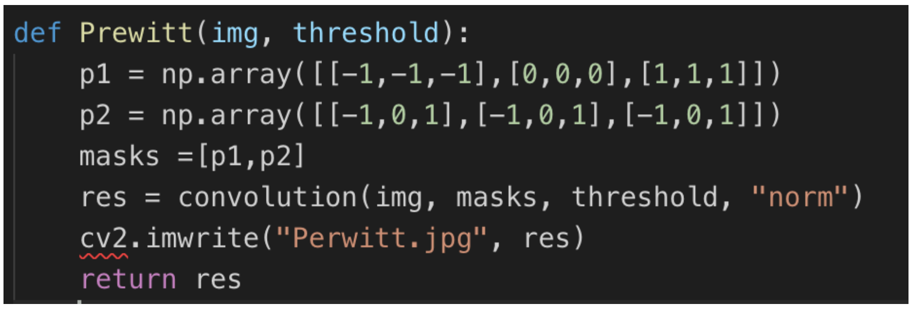


```python
def Robert(img, threshold):
    r1 = np.array([[-1,0],[0,1]])
    r2 = np.array([[0,-1],[1,0]])
    masks = [r1,r2]
    res = convolution(img, masks, threshold, "norm")
    cv2.imwrite("Robert.jpg", res)
    return res
```

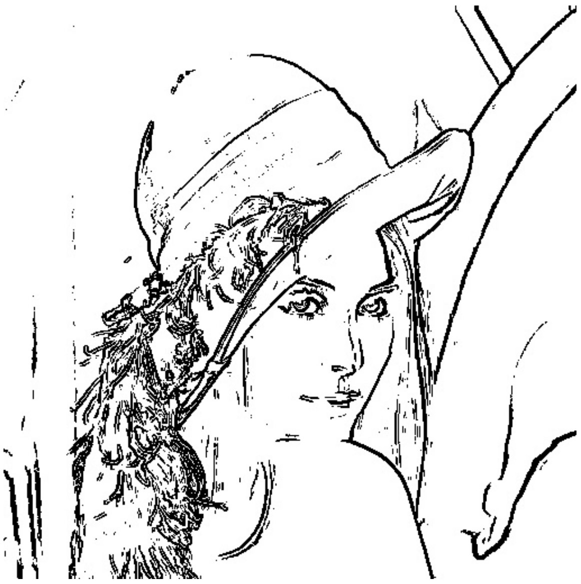


Prewitt's Edge Detector

Threshold = 80
Prewitt Function Code

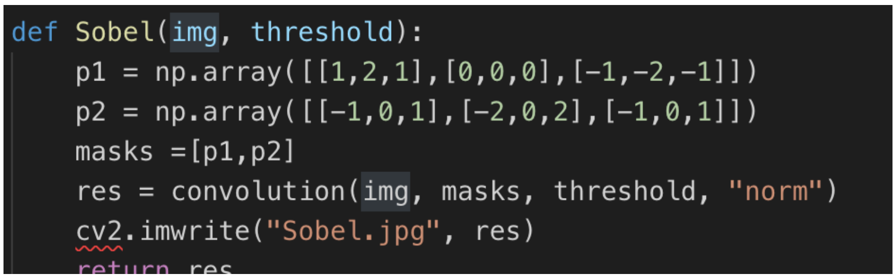


```python
def Prewitt(img, threshold):
    p1 = np.array([[-1,-1,-1],[0,0,0],[1,1,1]])
    p2 = np.array([[-1,0,1],[-1,0,1],[-1,0,1]])
    masks =[p1,p2]
    res = convolution(img, masks, threshold, "norm")
    cv2.imwrite("Perwitt.jpg", res)
    return res
```

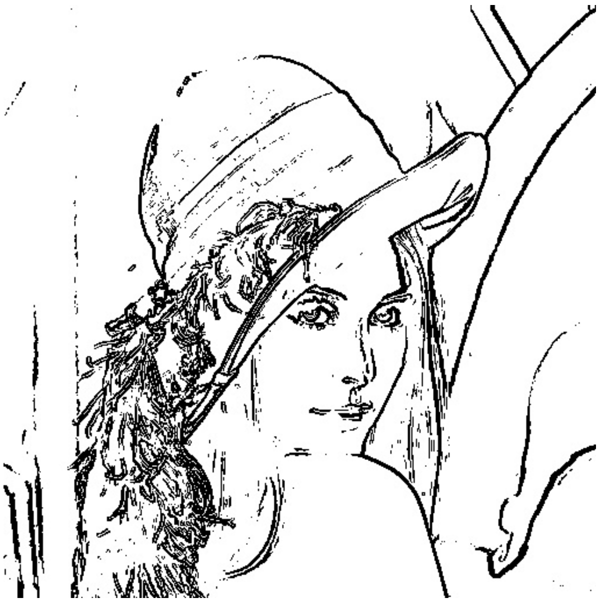


Sobel's Edge Detector

Threshold = 100
Sobel Function Code

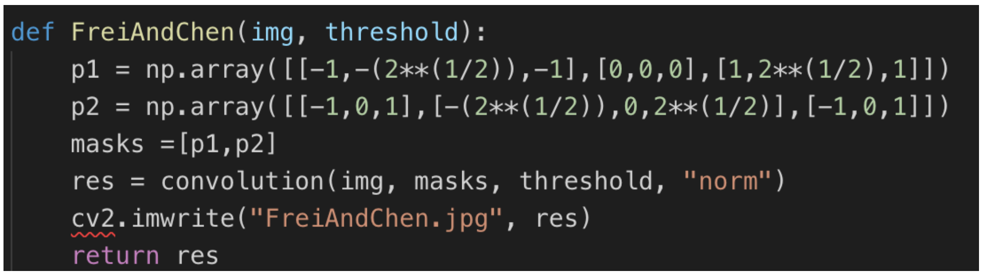


```
def Sobel(img, threshold):
    p1 = np.array([[1,2,1],[0,0,0],[-1,-2,-1]])
    p2 = np.array([[-1,0,1],[-2,0,2],[-1,0,1]])
    masks =[p1,p2]
    res = convolution(img, masks, threshold, "norm")
    cv2.imwrite("Sobel.jpg", res)
    return res
```

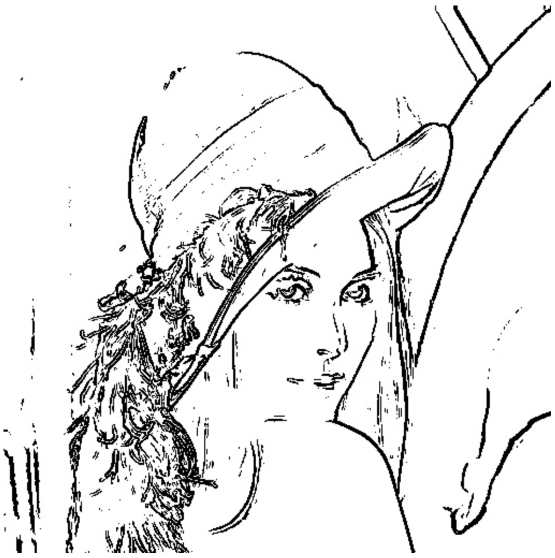


Frei and Chen's Gradient Operator

Threshold = 100
Frei and Chen Function Code



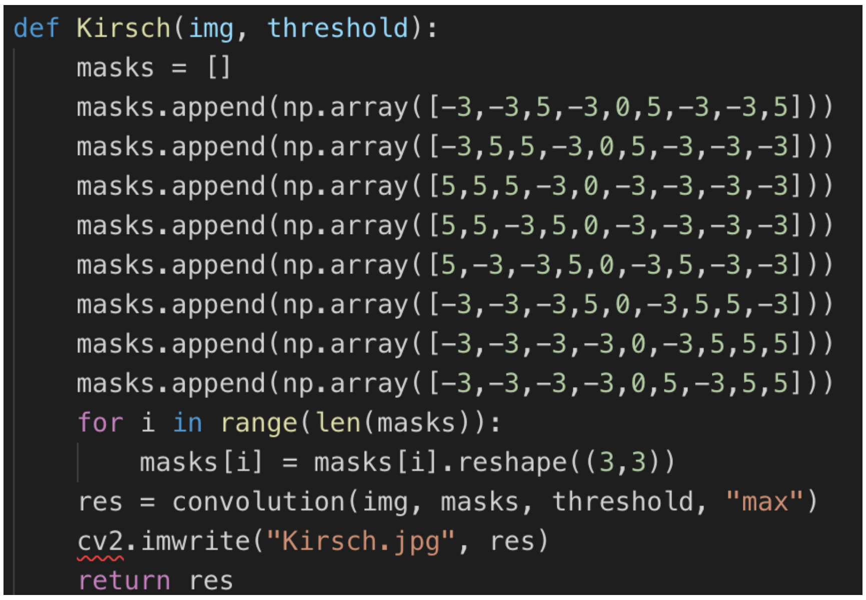
```
def FreiAndChen(img, threshold):
    p1 = np.array([[-1,-(2**(1/2)),-1],[0,0,0],[1,2**(1/2),1]])
    p2 = np.array([[-1,0,1],[-(2**(1/2)),0,2**(1/2)],[-1,0,1]])
    masks =[p1,p2]
    res = convolution(img, masks, threshold, "norm")
    cv2.imwrite("FreiAndChen.jpg", res)
    return res
```



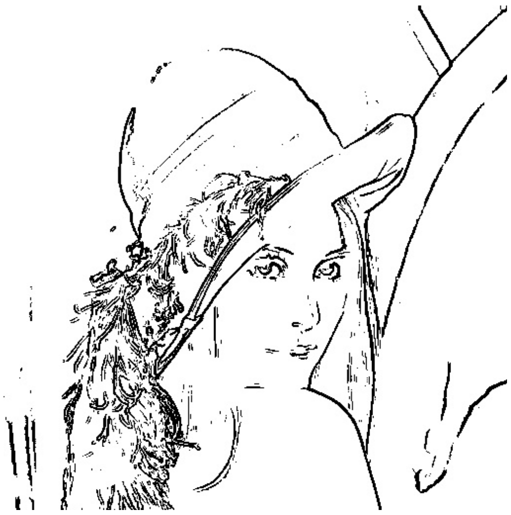
Kirsch's Compass Operator

Threshold = 400

Kirsch's Function Code

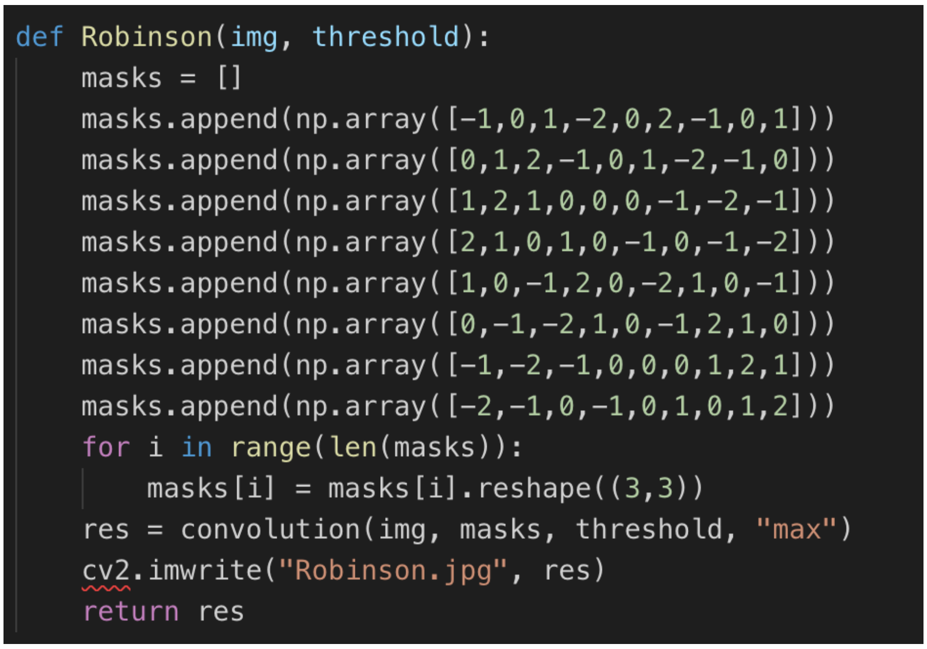


```python
def Kirsch(img, threshold):
    masks = []
    masks.append(np.array([-3,-3,5,-3,0,5,-3,-3,5]))
    masks.append(np.array([-3,5,5,-3,0,5,-3,-3,-3]))
    masks.append(np.array([5,5,5,-3,0,-3,-3,-3,-3]))
    masks.append(np.array([5,5,-3,5,0,-3,-3,-3,-3]))
    masks.append(np.array([5,-3,-3,5,0,-3,5,-3,-3]))
    masks.append(np.array([-3,-3,-3,5,0,-3,5,5,-3]))
    masks.append(np.array([-3,-3,-3,-3,0,-3,5,5,5]))
    masks.append(np.array([-3,-3,-3,-3,0,5,-3,5,5]))
    for i in range(len(masks)):
        masks[i] = masks[i].reshape((3,3))
    res = convolution(img, masks, threshold, "max")
    cv2.imwrite("Kirsch.jpg", res)
    return res
```

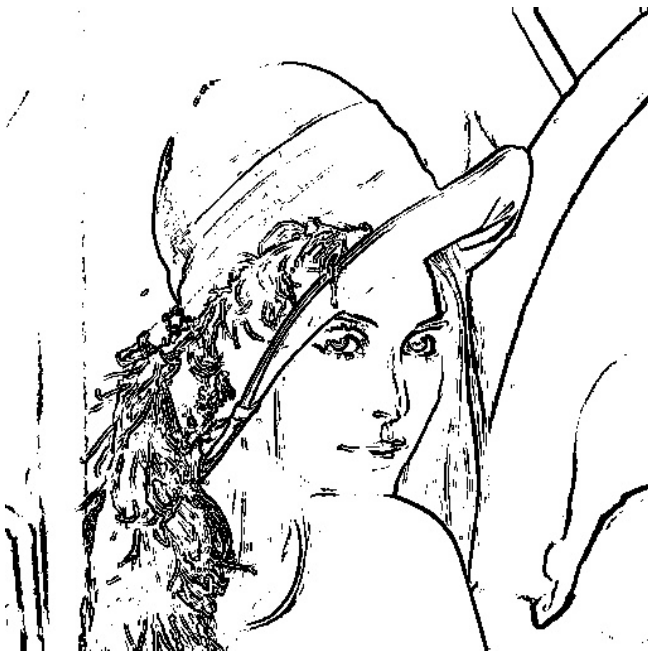


Robinson's Compass Operator

Threshold = 100
Robinson's Function Code

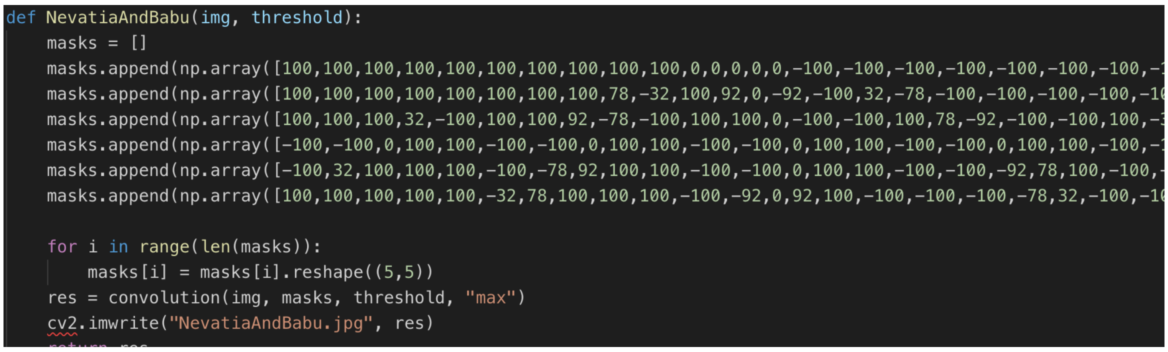


```python
def Robinson(img, threshold):
    masks = []
    masks.append(np.array([-1,0,1,-2,0,2,-1,0,1]))
    masks.append(np.array([0,1,2,-1,0,1,-2,-1,0]))
    masks.append(np.array([1,2,1,0,0,0,-1,-2,-1]))
    masks.append(np.array([2,1,0,1,0,-1,0,-1,-2]))
    masks.append(np.array([1,0,-1,2,0,-2,1,0,-1]))
    masks.append(np.array([0,-1,-2,1,0,-1,2,1,0]))
    masks.append(np.array([-1,-2,-1,0,0,0,1,2,1]))
    masks.append(np.array([-2,-1,0,-1,0,1,0,1,2]))
    for i in range(len(masks)):
        masks[i] = masks[i].reshape((3,3))
    res = convolution(img, masks, threshold, "max")
    cv2.imwrite("Robinson.jpg", res)
    return res
```



Nevatia-Babu 5x5 Operator

Threshold = 30000
Nevatia-Babu Function Code



```python
def NevatiaAndBabu(img, threshold):
    masks = []
    masks.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,-
    masks.append(np.array([100,100,100,100,100,100,100,100,78,-32,100,92,0,-92,-100,32,-78,-100,-100,-100,-100,-1
    masks.append(np.array([100,100,100,32,-100,100,100,92,-78,-100,100,100,0,-100,-100,100,78,-92,-100,-100,100,-
    masks.append(np.array([-100,-100,0,100,100,-100,-100,0,100,100,-100,-100,0,100,100,-100,-100,0,100,100,-100,-
    masks.append(np.array([-100,32,100,100,100,-100,-78,92,100,100,-100,-100,0,100,100,-100,-100,-92,78,100,-100,
    masks.append(np.array([100,100,100,100,100,-32,78,100,100,100,-100,-92,0,92,100,-100,-100,-100,-78,32,-100,-1

    for i in range(len(masks)):
        masks[i] = masks[i].reshape((5,5))
    res = convolution(img, masks, threshold, "max")
    cv2.imwrite("NevatiaAndBabu.jpg", res)
    return res
```