電腦視覺 Homework 9 – General Edge Detection

資工碩二 r08922a04 林承德

Language : python 3.7
Library : Numpy, opencv(cv2)
Execution way: python3 hw9.py
(please put lena.bmp at the same directory with hw9.py)

Operators	Threshold	Result	Description
Robert's Operator	12		讀入照片後，利用先將其向外做padding (上下左右各padding 一個pixel)，再依照PPT中的filter取值，計算gradient，結果如左，code如下

```python
def Roberts(self, image, threshold):
    res = np.ones((512, 512), dtype = np.uint8)
    h, w = image.shape
    for i in range(1, h-1):
        for j in range(1, w-1):
            v1 = int(image[i+1, j+1]) - int(image[i, j])
            v2 = int(image[i+1, j]) - int(image[i, j+1])
            res[i-1][j-1] = self.gradient(v1, v2, threshold)

    return res
```

Prewitt's Edge Detector	24		讀入照片後，跟先前一樣，先對突變的上下左右各padding一個pixel，接著也是依照PPt中的filter去取值，計算gradient，結果如左，code如下

```python
def Prewitt(self, image, threshold):
    res = np.ones((512, 512), dtype = np.uint8)
    h, w = image.shape
    for i in range(1, h-1):
        for j in range(1, w-1):
            v1 = sum(image[i+1, j-1:j+2]) - sum(image[i-1, j-1:j+2])
            v2 = sum(image[i-1:i+2, j+1]) - sum(image[i-1:i+2, j-1])
            res[i-1][j-1] = self.gradient(v1, v2, threshold)
    return res
```

Sobel's Edge Detector	38		與前兩者相同，輸入照片後先對其進行padding，得到padding後的image，再依據PPT上所附的filter，進行相應運算，以及gradient的計算，結果如左圖，code如下

```python
def Sobel(self, image, threshold):
    res = np.zeros((512, 512), dtype = np.uint8)
    h, w = image.shape
    for i in range(1, h-1):
        for j in range(1, w-1):
            v1 = (image[i+1][j-1] + 2*image[i+1][j] + image[i+1][j+1]) - (image[i-1][j-1] + 2*image[i-1][j] + image[i-1][j+1])
            v2 = (image[i-1][j+1] + 2*image[i][j+1] + image[i+1][j+1]) - (image[i-1][j-1] + 2*image[i][j-1] + image[i+1][j-1])
            res[i-1][j-1] = self.gradient(v1, v2, threshold)
    return res
```

Frei and Chen's Gradient Operator	30		將padding後的image讀入，根據PPT所給的公式進行filter運算，將每個pixel的結果送進gradient function計算後可以更新pixel。最後結果如左，code如下

```
def FreiAndChen(self, image, threshold):
    res = np.ones((512, 512), dtype = np.uint8)
    h, w = image.shape
    for i in range(1, h-1):
        for j in range(1, w-1):
            v1 = (image[i+1][j-1] + 2**0.5 * image[i+1][j] + image[i+1][j+1]) - (image[i-1][j-1] + 2**0.5 * image[i-1][j] + image[i-1][j+1])
            v2 = (image[i-1][j+1] + 2**0.5 * image[i][j+1] + image[i+1][j+1]) - (image[i-1][j-1] + 2**0.5 * image[i][j-1] + image[i+1][j-1])
            res[i-1][j-1] = self.gradient(v1, v2, threshold)
    return res
```

Kirsch's Compass Operator	135		將padding後的照片與threshold輸入，用3個for迴圈去針對每個pixel，即每個filter，最後利 gradient_max function 可得該pixel的結果，最後結果如左，code如下

```
def Kirsch(self, image, threshold, filters):
    res = np.ones((512, 512), dtype = np.uint8)
    h, w = image.shape
    for i in range(1, h-1):
        for j in range(1, w-1):
            can = []
            temp = image[i-1:i+2, j-1:j+2]
            for f in filters:
                can.append(np.sum(temp*f))
            res[i-1][j-1] = self.gradient_max(can, threshold)
    return res
```

Robinson's Compass Operator	43		先將圖片做padding後，跟threshold一銅傳入，利用三個for迴圈分別走遍每個pixel與每個filter，將每個filter的結果放到一個list裡，送進gradient max計算後可得結果，最終結果如左，其code如下

```
def Robinson(self, image, threshold, filters):
    res = np.ones((512, 512), dtype = np.uint8)
    h, w = image.shape
    for i in range(1, h-1):
        for j in range(1, w-1):
            can = []
            for f in filters:
                can.append(np.sum(image[i-1:i+2, j-1:j+2] * f ))
            res[i-1][j-1] = self.gradient_max(can, threshold)
    return res
```

Nevatia-Babu 5x5 Operator	12500		吃進圖片後，對其上下左右分別padding兩個pixel後與threshold及filters輸入，同樣將將對同個pixel不同filter運算的結果 送入gradient max function，最後結果如左，code如下

```python
def NevatiaBabu(self, image, threshold, filters):
    res = np.ones((512, 512), dtype = np.uint8)
    h, w = image.shape
    for i in range(2, h-2):
        for j in range(2, w-2):
            can = []
            for f in filters:
                can.append(np.sum(image[i-2:i+3, j-2:j+3] * f))
            res[i-2][j-2] = self.gradient_max(can, threshold)
    return res
```

Overall Source Code

```python
class Solution:
    def gradient(self, v1, v2, threshold): ...

    def Roberts(self, image, threshold): ...

    def Prewitt(self, image, threshold): ...

    def Sobel(self, image, threshold): ...

    def FreiAndChen(self, image, threshold): ...

    def gradient_max(self, values, threshold): ...

    def Kirsch(self, image, threshold, filters): ...

    def Robinson(self, image, threshold, filters): ...

    def NevatiaBabu(self, image, threshold, filters): ...

    def read_image(self, lena_path):
        image = cv2.imread(lena_path, cv2.IMREAD_GRAYSCALE).astype(int)
        border1 = cv2.copyMakeBorder(image, 1, 1, 1, 1, cv2.BORDER_REPLICATE)
        border2 = cv2.copyMakeBorder(image, 2, 2, 2, 2, cv2.BORDER_REPLICATE)

        return image, border1, border2


    def save_image(self, image, path):
        cv2.imwrite(path, image)



def hw9(kirsch_f, robinson_f, babu_f, lena_path):
    # image = Image.open(lena_path)
    sol = Solution()
    image, border1, border2 = sol.read_image(lena_path)

    print('robert')
    roberts = sol.Roberts(border1, 12)
    sol.save_image(roberts, 'roberts.bmp')

    print('prewitt')
    prewitt = sol.Prewitt(border1, 24)
    sol.save_image(prewitt, 'prewitt.bmp')

    print('sobel')
    sobel = sol.Sobel(border1, 38)
    sol.save_image(sobel, 'sobel.bmp')

    print('fnc')
    fnc = sol.FreiAndChen(border1, 30)
    sol.save_image(fnc, 'fnc.bmp')

    print('kirsch')
    kirsch = sol.Kirsch(border1, 135, kirsch_f)
    sol.save_image(kirsch, 'kirsch.bmp')

    print('robinson')
    robinson = sol.Robinson(border1, 43, robinson_f)
    sol.save_image(robinson, 'robinson.bmp')

    print('babu')
    nev_babu = sol.NevatiaBabu(border2, 12500, babu_f)
    sol.save_image(nev_babu, 'nevatia_babu.bmp')


if __name__ == '__main__':

    path = 'lena.bmp'

    kirsch_f = np.array([[[-3, -3, 5], ...

    robinson_f = np.array([[[-1, 0, 1], ...

    babu_f = np.array([[[100, 100, 100, 100, 100], ...

    hw9(kirsch_f, robinson_f, babu_f, path)
```