Computer Vision I _2018

Homework assignment #9

R07522717 機械所製造組碩一 林温雅

```
#使用 python
#import 套件
import cv2
import numpy as np

original_img = cv2.imread('lena.bmp', 0)

def roberts(img):
    ker_r1 = np.array([[-1, 0],[0, 1]])
    ker_r2 = np.array([[0, -1],[1, 0]])
    rows, cols = img.shape
    # for center 在左上角
    temp_img = cv2.copyMakeBorder(src=img, top=0, bottom=1, left=0,
right=1, borderType=cv2.BORDER_REPLICATE)
    new_img = img.copy().astype(float)
    for i in range(rows):
        for j in range(cols):
            temp = temp_img[i:i+2, j:j+2]

            new_img[i,j] = np.sqrt(np.sum(np.multiply(ker_r1, temp))**2
+ np.sum(np.multiply(ker_r2, temp))**2)
            #new_img[i,j] = np.abs(np.sum(np.multiply(ker_r1, temp)))
+ np.abs(np.sum(np.multiply(ker_r2, temp)))

    return new_img

def perwitt(img):
    ker_p1 = np.array([[-1,-1,-1], [0,0,0], [1,1,1]])
    ker_p2 = np.array([[-1,0,1], [-1,0,1], [-1,0,1]])
    rows, cols = img.shape
    # for center 在左上角
    temp_img = cv2.copyMakeBorder(src=img, top=1, bottom=1, left=1,
right=1, borderType=cv2.BORDER_REPLICATE)
    new_img = img.copy().astype(float)
    for i in range(rows):
        for j in range(cols):
            temp = temp_img[i:i+3, j:j+3]
            #  平方相加開根號
            new_img[i, j] = np.sqrt(np.sum(np.multiply(ker_p1, temp)) **
2 + np.sum(np.multiply(ker_p2, temp)) ** 2)
            # abs???
    return new_img

def sobel(img):
    ker_p1 = np.array([[-1,-2,-1], [0,0,0], [1,2,1]])
    ker_p2 = np.array([[-1,0,1], [-2,0,2], [-1,0,1]])
    rows, cols = img.shape
    temp_img = cv2.copyMakeBorder(src=img, top=1, bottom=1, left=1,
right=1, borderType=cv2.BORDER_REPLICATE)
    new_img = img.copy().astype(float)
    for i in range(rows):
        for j in range(cols):
            temp = temp_img[i:i+3, j:j+3]
            #  平方相加開根號
            new_img[i, j] = np.sqrt(np.sum(np.multiply(ker_p1,
temp))**2 + np.sum(np.multiply(ker_p2, temp))**2)
    return new_img

def frei_chen(img):
    ker_p1 = np.array([[-1,-np.sqrt(2),-1], [0,0,0], [1,np.sqrt(2),1]])
    ker_p2 = np.array([[-1,0,1], [-np.sqrt(2),0,np.sqrt(2)], [-1,0,1]])
    rows, cols = img.shape
    temp_img = cv2.copyMakeBorder(src=img, top=1, bottom=1, left=1,
right=1, borderType=cv2.BORDER_REPLICATE)
    new_img = img.copy().astype(float)
    for i in range(rows):
        for j in range(cols):
            temp = temp_img[i:i+3, j:j+3]
            # 平方相加開根號
            new_img[i, j] = np.sqrt(np.sum(np.multiply(ker_p1,
temp))**2 + np.sum(np.multiply(ker_p2, temp))**2)
    return new_img

def krisch(img):
    ker_k0 = np.array([[-3,-3,5], [-3,0,5], [-3,-3,5]])
    ker_k1 = np.array([[-3,5,5], [-3,0,5], [-3,-3,-3]])
    ker_k2 = np.array([[5,5,5], [-3,0,-3], [-3,-3,-3]])
    ker_k3 = np.array([[5,5,-3], [5,0,-3], [-3,-3,-3]])
    ker_k4 = np.array([[5,-3,-3], [5,0,-3], [5,-3,-3]])
    ker_k5 = np.array([[-3,-3,-3], [5,0,-3], [5,5,-3]])
    ker_k6 = np.array([[-3,-3,-3], [-3,0,-3], [5,5,5]])
    ker_k7 = np.array([[-3,-3,-3], [-3,0,5], [-3,5,5]])
    list_kn = [ker_k0, ker_k1, ker_k2, ker_k3, ker_k4, ker_k5, ker_k6, ker_k7]
    rows, cols = img.shape
    temp_img = cv2.copyMakeBorder(src=img, top=1, bottom=1, left=1,
right=1, borderType=cv2.BORDER_REPLICATE)
    new_img = img.copy().astype(float)
    for i in range(rows):
        for j in range(cols):
            temp = temp_img[i:i+3, j:j+3]
            max=0 # 初始化 max 值
            for ker in list_kn:
                temp_sum = np.sum(ker * temp)
                if temp_sum > max:
                    max = temp_sum
            new_img[i, j] = max

    return new_img

def robinson(img):
    ker_r0 = np.array([[-1,0,1], [-2,0,2], [-1,0,1]])
    ker_r1 = np.array([[0,1,2], [-1,0,1], [-2,-1,0]])
    ker_r2 = np.array([[1,2,1], [0,0,0], [-1,-2,-1]])
    ker_r3 = np.array([[2,1,0], [1,0,-1], [0,-1,-2]])
    ker_r4 = np.array([[1,0,-1], [2,0,-2], [1,0,-1]])
    ker_r5 = np.array([[0,-1,-2], [1,0,-1], [2,1,0]])
    ker_r6 = np.array([[-1,-2,-1], [0,0,0], [1,2,1]])
    ker_r7 = np.array([[-2,-1,0], [-1,0,1], [0,1,2]])
    list_rn = [ker_r0, ker_r1, ker_r2, ker_r3, ker_r4, ker_r5, ker_r6, ker_r7]
    rows, cols = img.shape
    temp_img = cv2.copyMakeBorder(src=img, top=1, bottom=1, left=1,
right=1, borderType=cv2.BORDER_REPLICATE)
    new_img = img.copy().astype(float)
    for i in range(rows):
        for j in range(cols):
            temp = temp_img[i:i+3, j:j+3]
            max=0 #  初始化 max 值
            for ker in list_rn:
                temp_sum = np.sum(ker * temp)
                if temp_sum > max:
                    max = temp_sum
            new_img[i, j] = max
    return new_img


def nevatia_babu(img):
    ker_nb0 = 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]])
    ker_nb1 = 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]])
    ker_nb2 = 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]])
    ker_nb3 = 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]])
    ker_nb4 = 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]])
    ker_nb5 = 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]])

    list_nbn = [ker_nb0, ker_nb1, ker_nb2, ker_nb3, ker_nb4, ker_nb5]
    rows, cols = img.shape
    temp_img = cv2.copyMakeBorder(src=img, top=2, bottom=2, left=2,
right=2, borderType=cv2.BORDER_REPLICATE)
    new_img = img.copy().astype(float)
    for i in range(rows):
        for j in range(cols):
            temp = temp_img[i:i+5, j:j+5]
            max=0 # 初始化 max 值
            for ker in list_nbn:
                temp_sum = np.sum(ker * temp)
                if temp_sum > max:
                    max = temp_sum
            new_img[i, j] = max
    return new_img



def reverse_thresholding(img, threshold=128):
    new_img = np.empty(img.shape)
    new_img.fill(255)
    mask = img >= threshold
    new_img[mask] = 0
    return  new_img

roberts_img = roberts(original_img)
perwitt_img = perwitt(original_img)
sobel_img = sobel(original_img)
frei_chen_img = frei_chen(original_img)
krisch_img = krisch(original_img)
robinson_img = robinson(original_img)
```

```
nevatia_babu_img = nevatia_babu(original_img)

cv2.imwrite('roberts_30.bmp', reverse_thresholding(roberts_img, 30))
cv2.imwrite('perwitt_90.bmp', reverse_thresholding(perwitt_img, 90))
cv2.imwrite('sobel_130.bmp', reverse_thresholding(sobel_img, 130))
cv2.imwrite('frei_chen_110.bmp', reverse_thresholding(frei_chen_img,
110))
cv2.imwrite('krisch_230.bmp', reverse_thresholding(krisch_img, 230))
cv2.imwrite('robinson_120.bmp', reverse_thresholding(robinson_img,
120))
cv2.imwrite('nevatia_babu_30000.bmp',
reverse_thresholding(nevatia_babu_img, 30000))
```