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))