Computer Vision I \_2018

Homework assignment #7

R07522717 機械所製造組碩一 林温雅

#使用python

#import套件

# -\*- coding: utf-8 -\*-  
  
*import* cv2  
*import* numpy *as* np  
  
# 讀取原始影像  
original\_img = cv2.imread('lena.bmp', 0)  
  
# 進行二值化用的function  
*def* Binarize(*img*):  
 rows, columns = *img*.shape  
 new\_img = np.zeros((rows, columns), np.int)  
 *for* i *in* range(rows):  
 *for* j *in* range(columns):  
 *if img*[i, j] >= 128:  
 new\_img[i, j] = 255  
 *return* new\_img  
  
  
# 進行DownSampling用的function  
*def* DownSampling(*img*, *scale*):  
 rows, columns = *img*.shape  
 new\_img = np.zeros((int(rows / *scale*), int(columns / *scale*)), np.int)  
 *for* i *in* range(0, rows, *scale*):  
 *for* j *in* range(0, columns, *scale*):  
 new\_img[int(i / *scale*), int(j / *scale*)] = *img*[i, j]  
  
 *return* new\_img  
  
# 進行Yokoi計算的輔助function，供YokoiNum function使用，使用4-connectivity  
*def* YokoiCalc(*b*, *c*, *d*, *e*):  
 *if b* == *c*:  
 *if* (*d* != *b*) *or* (*e* != *b*):  
 *return* 'q'  
 *elif* (*d* == *b*) *and* (*e* == *b*):  
 *return* 'r'  
 *elif b* != *c*:  
 *return* 's'  
  
*def* Yokoi\_Single\_Point(*img*, *i*, *j*):  
 # 獲得輸入圖檔之行列數  
 rows, columns = *img*.shape  
 # 擴大圖檔每邊各一條  
 temp\_img = np.zeros((rows + 2, columns + 2), np.int)  
 temp\_img[1:rows + 1, 1:columns + 1] = *img* # 製作一個新圖檔準備接受處理後的圖  
 new\_img = np.zeros((rows, columns), np.int)  
 dict\_f = dict({'q': 0, 's': 0, 'r': 0})  
 # dict\_f['q'], dict\_f['s'], dict\_f['r'] = 0, 0, 0  
 *i* += 1  
 *j* += 1  
 dict\_f[YokoiCalc(temp\_img[*i*, *j*], temp\_img[*i*, *j* + 1], temp\_img[*i* - 1, *j* + 1], temp\_img[*i* - 1, *j*])] += 1  
 dict\_f[YokoiCalc(temp\_img[*i*, *j*], temp\_img[*i* - 1, *j*], temp\_img[*i* - 1, *j* - 1], temp\_img[*i*, *j* - 1])] += 1  
 dict\_f[YokoiCalc(temp\_img[*i*, *j*], temp\_img[*i*, *j* - 1], temp\_img[*i* + 1, *j* - 1], temp\_img[*i* + 1, *j*])] += 1  
 dict\_f[YokoiCalc(temp\_img[*i*, *j*], temp\_img[*i* + 1, *j*], temp\_img[*i* + 1, *j* + 1], temp\_img[*i*, *j* + 1])] += 1  
 *if* dict\_f['r'] *is* 4:  
 *return* 5  
 *else*:  
 *return* dict\_f['q']  
  
  
# 對整張圖檔進行Yokoi計算的function(4,8通用)  
*def* YokoiNum(*img*):  
 # 獲得輸入圖檔之行列數  
 rows, columns = *img*.shape  
 # 擴大圖檔每邊各一條  
 temp\_img = np.zeros((rows + 2, columns + 2), np.int)  
 temp\_img[1:rows + 1, 1:columns + 1] = *img*.copy()  
 # 製作一個新圖檔準備接受處理後的圖  
 new\_img = np.zeros((rows, columns), np.int)  
 dict\_f = dict({'q': 0, 's': 0, 'r': 0})  
 *for* i *in* range(1, 1 + rows):  
 *for* j *in* range(1, 1 + columns):  
 *if* temp\_img[i, j] != 255:  
 *continue* dict\_f['q'], dict\_f['s'], dict\_f['r'] = 0, 0, 0  
 dict\_f[YokoiCalc(temp\_img[i, j], temp\_img[i, j + 1], temp\_img[i - 1, j + 1], temp\_img[i - 1, j])] += 1  
 dict\_f[YokoiCalc(temp\_img[i, j], temp\_img[i - 1, j], temp\_img[i - 1, j - 1], temp\_img[i, j - 1])] += 1  
 dict\_f[YokoiCalc(temp\_img[i, j], temp\_img[i, j - 1], temp\_img[i + 1, j - 1], temp\_img[i + 1, j])] += 1  
 dict\_f[YokoiCalc(temp\_img[i, j], temp\_img[i + 1, j], temp\_img[i + 1, j + 1], temp\_img[i, j + 1])] += 1  
  
 *if* dict\_f['r'] *is* 4:  
 new\_img[i - 1, j - 1] = 5  
 *else*:  
 new\_img[i - 1, j - 1] = dict\_f['q']  
 *return* new\_img  
  
  
# 如果yokoi是1，就可以砍(removable)，要input的是原始binary image  
*def* Connected\_Shrink(*img*):  
 # 用一個boolean array，是removable的就是True，其他是False  
 new\_img = np.full(*img*.shape, False, dtype=bool)  
 temp\_img = YokoiNum(*img*)  
 # 獲得輸入圖檔之行列數  
 rows, columns = *img*.shape  
 *for* i *in* range(rows):  
 *for* j *in* range(columns):  
 *if* temp\_img[i, j] == 1: # or temp\_img[i, j] == 0:  
 new\_img[i, j] = True  
  
 *return* new\_img  
  
#把p設為true, q設為false  
*def* Marked(*img*):  
 rows, columns = *img*.shape  
 # new\_img = np.chararray(img.shape, unicode=True)  
 #temp\_img = np.chararray((rows + 2, columns + 2), unicode=True)  
 temp\_img = np.zeros((rows + 2, columns + 2), np.int)  
 temp\_img[1:rows + 1, 1:columns + 1] = *img*.copy()  
  
 # new一個boolean array，要mark的就是True，其他是False  
 new\_img = np.full(*img*.shape, False, dtype=bool)  
 *for* i *in* range(1, rows + 1):  
 *for* j *in* range(1, columns + 1):  
 *if* temp\_img[i, j] == 1:  
 templist = [temp\_img[i][j+1], temp\_img[i-1][j], temp\_img[i][j-1], temp\_img[i+1][j]]  
 *if* 1 *in* templist:  
 new\_img[i - 1, j - 1] = True  
 *return* new\_img  
  
# 將圖檔二值化  
binarize\_lena = Binarize(original\_img)  
# 將二值化之圖檔進行邊長8倍的downscaling  
downsampling\_lena = DownSampling(binarize\_lena, 8)  
processed\_original\_img = downsampling\_lena.copy()  
final\_img = processed\_original\_img.copy()  
  
*while* True:  
 anythingchanged = False  
 # yokoi數字圖  
 # 本身是1，而且4-connected周邊至少也有一個1  
 yokoi = YokoiNum(processed\_original\_img)  
 marked\_img = Marked(yokoi)  
 *for* i *in* range(64):  
 *for* j *in* range(64):  
 *if* Yokoi\_Single\_Point(processed\_original\_img, i, j) == 1 *and* marked\_img[i, j]:  
 final\_img[i, j] = 0  
 processed\_original\_img = final\_img.copy()  
 anythingchanged = True  
 *if not* anythingchanged:  
 *break  
 else*:  
 processed\_original\_img = final\_img.copy()  
  
  
cv2.imwrite('thin\_lena.bmp', final\_img)