Computer Vision I \_2018

Homework assignment #4

R07522717 機械所製造組碩一 林温雅

#使用python

#import套件

import cv2

import numpy as np

#dilation function

def dilation(img, ker):

#獲得輸入圖檔之行列數

img\_rows, img\_columns = img.shape

#獲得kernel之行列數

ker\_rows, ker\_columns = ker.shape

#計算kernel中心距離邊界有多遠，為的是擴大原始圖檔，方便後續迴圈處理

row\_dist, column\_dist = int((ker\_rows-1)/2), int((ker\_columns-1)/2)

#根據上述計算，製作一個比原始圖檔大的暫存圖檔，以img為512\*512, kernel為5\*5來說，暫存圖檔為516\*516，暫存圖檔為往上、往下、往左、往右分別外擴兩列/行，外擴新增的pixel值為0，中間則就是原本輸入圖檔的值

temp\_img = np.zeros((img\_rows+2\*row\_dist, img\_columns+2\*column\_dist), np.int)

temp\_img[row\_dist:img\_rows+row\_dist, column\_dist:img\_columns+column\_dist] = img

#製作一個新圖檔準備接受dilation後的圖

new\_img = np.zeros((img\_rows, img\_columns), np.int)

#進行dilation邏輯計算

for i in range(row\_dist, img\_rows+row\_dist):

for j in range(column\_dist, img\_columns+column\_dist):

if np.any(np.logical\_and(ker, temp\_img[i-row\_dist: i+row\_dist+1, j-column\_dist: j+column\_dist+1])):

new\_img[i-row\_dist, j-column\_dist] = 255

return new\_img

def erosion(img, ker):

#獲得輸入圖檔之行列數

img\_rows, img\_columns = img.shape

#獲得kernel之行列數

ker\_rows, ker\_columns = ker.shape

#計算kernel中心距離邊界有多遠，為的是擴大原始圖檔，方便後續迴圈處理

row\_dist, column\_dist = int((ker\_rows-1)/2), int((ker\_columns-1)/2)

#根據上述計算，製作一個比原始圖檔大的暫存圖檔，以img為512\*512, kernel為5\*5來說，暫存圖檔為516\*516，暫存圖檔為往上、往下、往左、往右分別外擴兩列/行，外擴新增的pixel值為1，中間則就是原本輸入圖檔的值

###特別注意這邊外擴pixel的值為1，而在dilation function內是0###

temp\_img = np.ones((img\_rows+2\*row\_dist, img\_columns+2\*column\_dist), np.int)

temp\_img[row\_dist:img\_rows+row\_dist, column\_dist:img\_columns+column\_dist] = img

#製作一個新圖檔準備接受dilation後的圖

new\_img = np.zeros((img\_rows, img\_columns), np.int)

#進行erosion邏輯計算

for i in range(row\_dist, img\_rows+row\_dist):

for j in range(column\_dist, img\_columns+column\_dist):

if not np.any(ker - np.logical\_and(ker, temp\_img[i-row\_dist: i+row\_dist+1, j-column\_dist: j+column\_dist+1])):

new\_img[i-row\_dist, j-column\_dist] = 255

return new\_img

#Opening，對影像進行Erosion後再Dilation

def opening(img, ker):

return dilation(erosion(img, ker), ker)

#Closing，對影像進行Dilation後再Erosion

def closing(img, ker):

return erosion(dilation(img, ker), ker)

#對binary image製作黑白互補圖用的function

def binary\_image\_complement(img):

img\_rows, img\_columns = img.shape

new\_img = np.zeros((img\_rows, img\_columns), np.int)

for i in range(img\_rows):

for j in range(img\_columns):

new\_img[i,j] = 255-img[i,j]

return new\_img

#尋找upper-right-corner用的hit\_and\_miss function

def hit\_and\_miss\_ur\_corner(img, ker\_j, ker\_k):

img\_rows, img\_columns = img.shape

new\_img = np.zeros((img\_rows, img\_columns), np.int)

temp\_img1 = erosion(img, ker\_j)

temp\_img2 = erosion(binary\_image\_complement(img), ker\_k)

for i in range(img\_rows):

for j in range(img\_columns):

if temp\_img1[i,j]==255 and temp\_img2[i,j]==255:

new\_img[i,j] = 255

return new\_img

###開始處理###

#讀取hw2實作之binary圖檔

original\_img = cv2.imread('128binary\_lena.bmp', 0)

###製作kernel###

#dilation, erosion, opening, closing要用的kernel

kernel = np.array([[0,1,1,1,0], [1,1,1,1,1], [1,1,1,1,1], [1,1,1,1,1], [0,1,1,1,0]])

#hit\_and\_miss要用的j和k kernel

kernel\_j = np.array([[0,0,0], [1,1,0], [0,1,0]])

kernel\_k = np.array([[0,1,1], [0,0,1], [0,0,0]])

###輸出圖片###

#輸出dilation圖片

cv2.imwrite('dilation\_lena.bmp', dilation(original\_img, kernel))

#輸出erosion圖片

cv2.imwrite('erosion\_lena.bmp', erosion(original\_img, kernel))

#輸出opening圖片

cv2.imwrite('opening\_lena.bmp', opening(original\_img, kernel))

#輸出closing圖片

cv2.imwrite('closing\_lena.bmp', closing(original\_img, kernel))

#輸出hit\_and\_miss右上corner圖片

cv2.imwrite('hit\_and\_miss\_ur\_corner.bmp', hit\_and\_miss\_ur\_corner(original\_img, kernel\_j, kernel\_k))