Computer Vision I \_2018

Homework assignment #5

R07522717 機械所製造組碩一 林温雅

#使用python

#import套件

import cv2

import numpy as np

def GrayScale\_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值另為0，中間則就是原本輸入圖檔的值

#dilation要找最大的，所以外擴的填0

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

#為了for迴圈裡面index好寫，這邊一樣把new\_img改成擴大後的，之後再來裁，和hw4做法有一點點不一樣

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

#為了矩陣相乘，先flip kernel，erosion不用這樣

kernel\_flip = np.flip(ker)

#進行dilation計算

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

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

new\_img[i, j] = np.nanmax(temp\_img[i-row\_dist: i+row\_dist+1, j-column\_dist: j+column\_dist+1]+kernel\_flip)

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

return new\_img

def GrayScale\_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值另為0，中間則就是原本輸入圖檔的值

#erosion要找最小的，所以外擴的填255

temp\_img = 255 \* 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

#為了for迴圈裡面index好寫，這邊一樣把new\_img改成擴大後的，之後再來裁，和hw4做法有一點點不一樣

new\_img = 255\*np.ones((img\_rows+2\*row\_dist, img\_columns+2\*column\_dist), np.int)

#進行erosion計算

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

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

new\_img[i, j] = np.nanmin(temp\_img[i-row\_dist: i+row\_dist+1, j-column\_dist: j+column\_dist+1]-ker)

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

return new\_img

def GrayScale\_Opening(img, ker):

return GrayScale\_Dilation(GrayScale\_Erosion(img, ker), ker)

def GrayScale\_Closing(img, ker):

return GrayScale\_Erosion(GrayScale\_Dilation(img, ker), ker)

original\_img = cv2.imread('lena.bmp', 0)

###製作kernel###

#dilation, erosion, opening, closing要用的kernel

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

###輸出圖片###

#輸出dilation圖片

cv2.imwrite('gray\_scale\_dilation\_lena.bmp', GrayScale\_Dilation(original\_img, kernel))

#輸出erosion圖片

cv2.imwrite('gray\_scale\_erosion\_lena.bmp', GrayScale\_Erosion(original\_img, kernel))

#輸出opening圖片

cv2.imwrite('gray\_scale\_opening\_lena.bmp', GrayScale\_Opening(original\_img, kernel))

#輸出closing圖片

cv2.imwrite('gray\_scale\_closing\_lena.bmp', GrayScale\_Closing(original\_img, kernel))