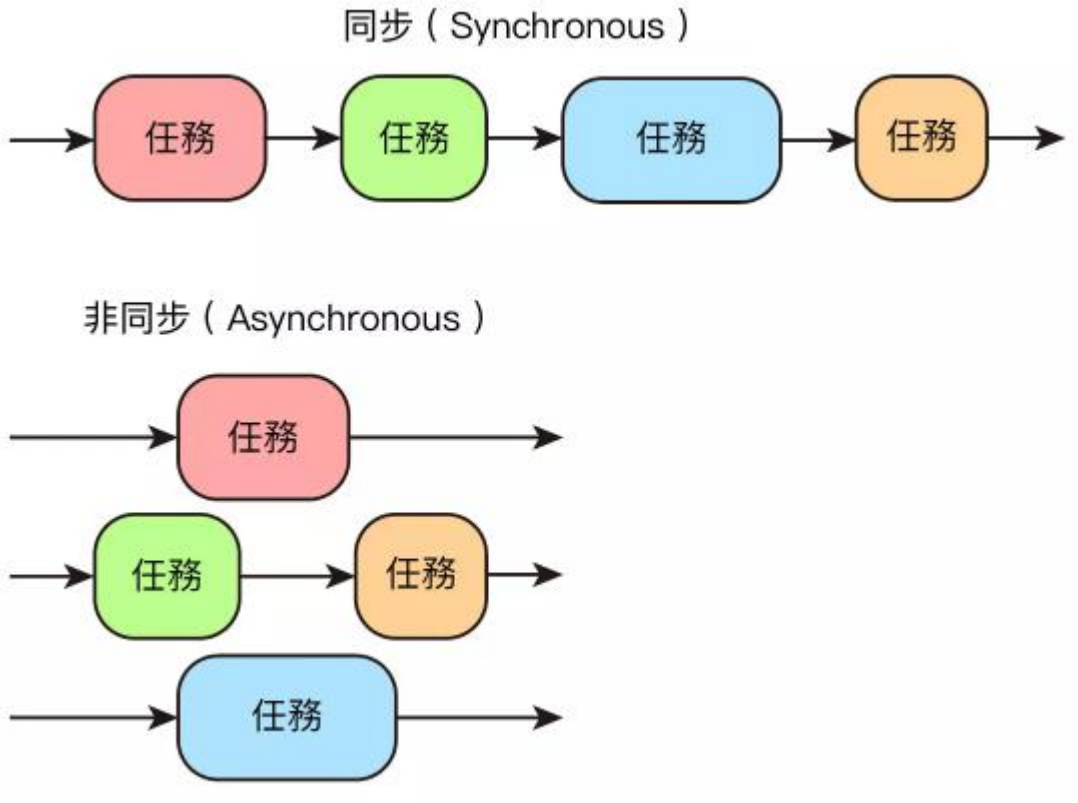
多媒體程式設計
影像資料處理

Instructor: 馬豪尚

Python任務處理流程

› Python 在執行時，通常是採用同步的任務處理模式，一個處理完成後才會接下去處理第二個



平行任務處理#1 concurrent.futures

- concurrent.futures標準函式提供了平行任務處理 (非同步) 的功能，能夠同時處理多個任務
  - ThreadPoolExecutor 針對 Thread (執行緒)
  - ProcessPoolExecutor 針對 Process (程序)

英文	中文	說明
Thread	執行緒	程式執行任務的基本單位。
Process	程序	啟動應用程式時產生的執行實體，需要一定的 CPU 與記憶體資源，Process 由一到多個 Thread 組成，同一個 Process 裡的 Thread 可以共用記憶體資源。

ThreadPoolExecutor

- 會透過 Thread 的方式建立多個 Executors (執行器)
- 執行並處理多個任務 (tasks)
- ThreadPoolExecutor 有四個參數

參數	說明
max_workers	Thread 的數量，預設 5 (CPU number * 5，每個 CPU 可以處理 5 個 Thread)，數量越多，運行速度會越快，如果設定小於等於 0 會發生錯誤。
thread_name_prefix	Thread 的名稱，預設 ''。
initializer	每個 Thread 啟動時調用的可調用對象，預設 None。
initargs	傳遞給初始化程序的參數，使用 tuple，預設 ()。

ThreadPoolExecutor

- 創建一個執行 Thread 的啟動器
  - executor = ThreadPoolExecutor()
- 使用 ThreadPoolExecutor 後，就能使用 Executors 的相關方法
  - executer.submit(fn, *args)

方法	參數	說明
submit	fn, *args, **kwargs	執行某個函式。
map	func, *iterables	使用 map 的方式，使用某個函式執行可迭代的內容。
shutdown	wait	完成執行後回傳信號，釋放正在使用的任何資源，wait 預設 True 會在所有對象完成後才回傳信號，wait 設定 False 則會在執行後立刻回傳。

ThreadPoolExecutor

- 迴圈架構可以用map來執行
- Example

```
with ThreadPoolExecutor() as executor:
    executor.submit(test, 2)
    executor.submit(test ,3)
    executor.submit(test, 4)
```

→

```
with ThreadPoolExecutor() as executor:
    executor.map(test, [2,3,4])
```

平行任務處理#2 threading

- 載入threading 多執行緒處理模組
  - import threading
- 建立 threading 的物件
  - thread = threading.Thread(target=function, args)
    - target=function為指定執行的函式
    - args為傳入函式的參數

Threading

› 建立 threading 物件之後，就可以使用下列常用的方法

方法	說明
start()	啟用執行緒。
join()	等待執行緒，直到該執行緒完成才會進行後續動作。
ident	取得該執行緒的標識符。
native_id	取得該執行緒的 id。
is_alive()	執行緒是否啟用，啟用 True，否則 False。

影像去除雜訊

影像雜訊

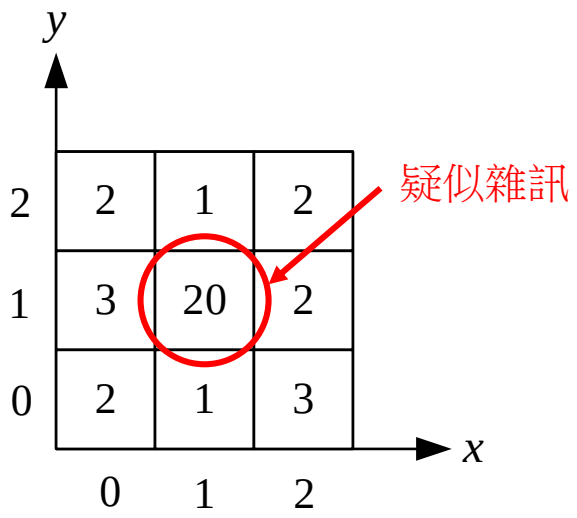
- 對於影像中的雜訊處，像素資訊已經被雜訊掩蓋而失真，若想移除雜訊從而還原影像，你只能試著在該處填個資訊

2	5	6	5
3	1	4	6
1	28	30	2
7	3	2	2

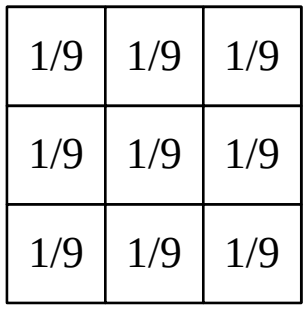
疑似雜訊

影像平滑模糊原理

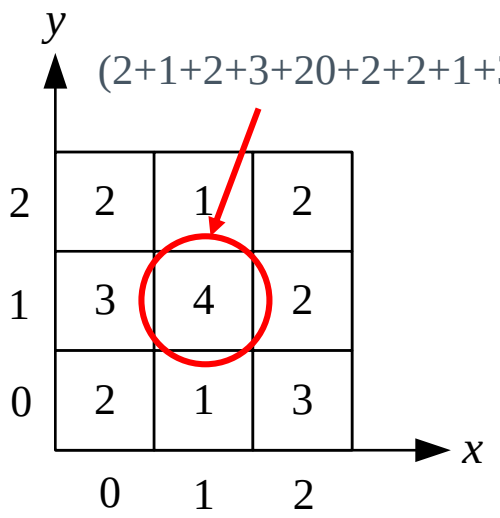
- 影像平滑(模糊)化是透過使用濾波器進行影像卷積來實現
- 從影像中去除高頻內容（例如，雜訊，邊緣），但也會導致影像邊緣變得模糊



3×3 子影像



濾波器遮罩



經平滑法作用於
中心點後的子影像

OpenCV影像模糊

- 平均模糊法
  - 平均濾波是使用濾波器遮罩進行影像卷積來完成
  - 簡單地計算 kernel 裡所有 pixel 的平均值，並將該平均值取代 kernel 中心元素
- cv2.blur(img, ksize)
  - img 為影像物件
  - ksize 為遮罩大小(X, X)
  - 遮罩設定的範圍越大，會讓影像變得越模糊

OpenCV影像模糊

- 高斯模糊
  - 平均濾波 Averaging 的 kernel 裡的每個 pixel 權重都是1
  - 高斯濾波給予每個 pixel 不同權重，中心 pixel 的權重最高，越往邊角權重就越低
  - 相較於平均濾波 Averaging 這樣可以讓圖片失真較少
- cv2.GaussianBlur(img, ksize, sigmaX, sigmaY)
  - img 為影像物件
  - Ksize 為遮罩大小(X, X)
  - sigmaX 為X 方向標準差，sigmaY 為Y 方向標準差，預設 0

OpenCV影像模糊

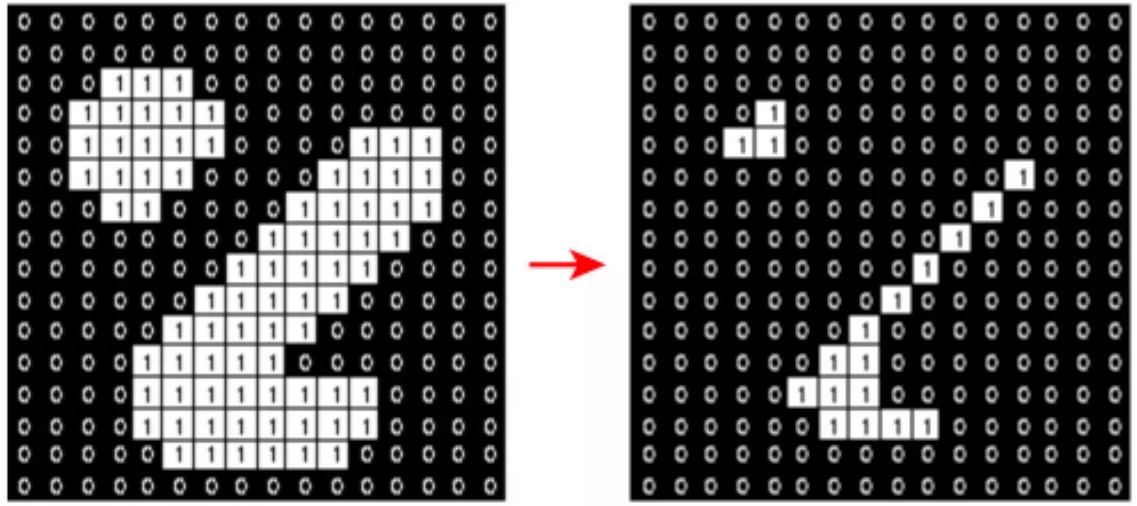
- 中值濾波
  - 計算遮罩內所有 pixel 的中位數然後取代 kernel 中間的數值
  - 中值濾波 Median Filtering 這個方法對於去除雜訊很有效
- cv2.medianBlur(img, ksize)
  - img 為影像物件
  - ksize 為遮罩大小 (必須是大於 1 的奇數)

OpenCV影像模糊

- 雙邊模糊
  - 透過非線性的雙邊濾波器進行計算
  - 雙邊濾波器除了使用像素之間幾何上的靠近程度之外，還多考慮了像素之間的光度/色彩差異
  - 影像模糊化的同時，也較能夠保留影像內容的邊緣
- cv2.bilateralFilter(img, d, sigmaColor, sigmaSpace)
  - img 為影像物件
  - d 為相鄰像素的直徑，輸入值為一個正整數
    - 數值越大運算的速度越慢
  - sigmaColor 為相鄰像素的顏色混合，輸入值為一個正整數
    - 數值越大，會混合更多區域的顏色，並產生更大區塊的同一種顏色
  - sigmaSpace 為定義會影響像素的區域，輸入值為一個正整數
    - 數值越大，影響的範圍就越大，影響的像素就越多

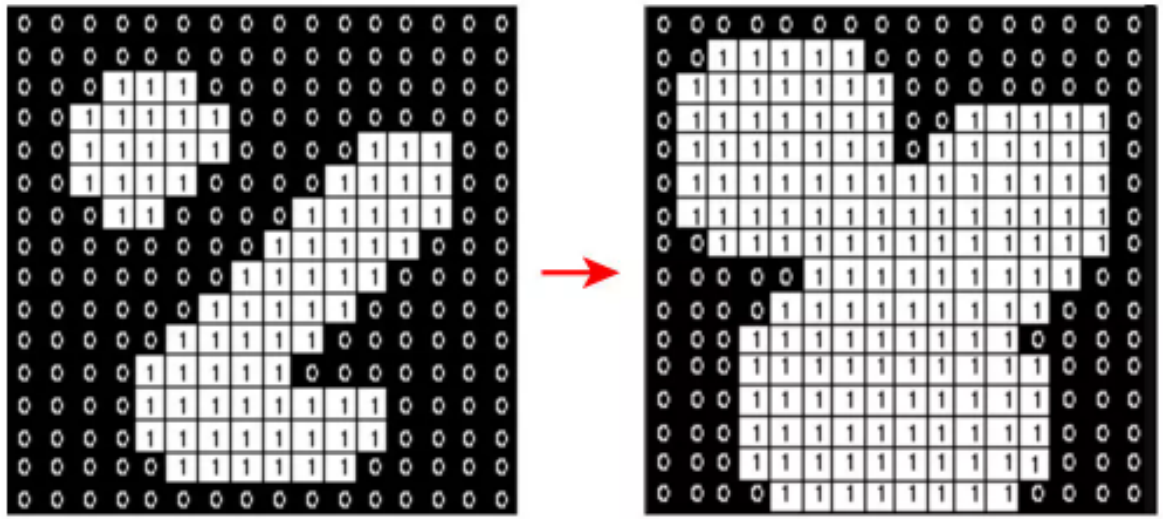
影像的侵蝕與膨脹

- 侵蝕 (Erosion)
  - 考慮在空間中的兩個集合A 集合和B 集合，當 A 集合的部分空間被 B 集合所取代，稱為A 集合被B 集合侵蝕
  - A集合通常為影像物件，而B集合為結構元素



影像的侵蝕與膨脹

- 膨脹 (Dilation)
  - 當空間中有兩個集合 (A 集合和 B 集合)，當 A 集合的部分空間擴張到 B 集合，稱之A集合膨脹，通常進行膨脹後的影像，影像看起來會擴大



影像的開放運算

- 先將圖片進行侵蝕，侵蝕後，比較小的白色圓點雜訊就會因為侵蝕而消失，接著再進行膨脹，就可以將主體結構恢復原本的大小，實現去除雜訊的效果
- 常搭配邊緣偵測、黑白二值化等方法，應用在文字辨識或影像辨識的領域

OpenCV影像的侵蝕與膨脹

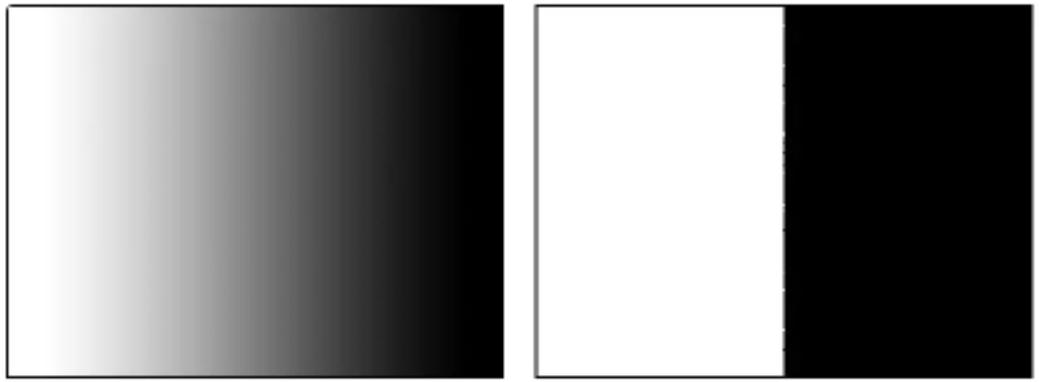
- 創建結構元素
  - cv2.getStructuringElement(shape, ksize)
  - 返回指定大小形狀的結構元素
  - shape 的值
    - cv2.MORPH_RECT (矩形)
    - cv2.MORPH_CROSS (十字交叉)
    - cv2.MORPH_ELLIPSE (橢圓形)
  - ksize 為結構元素的大小，輸入格式為(x, y)

OpenCV影像的侵蝕與膨脹

- 侵蝕
  - cv2.erode(img, kernel)
    - Img為影像物件
    - Kernel為結構元素物件
- 膨脹
  - cv2.dilate(img, kernel)
    - Img為影像物件
    - Kernel為結構元素物件

影像二值化

- 二值化又稱為「閾值二進制」，是一種簡單的圖像分割方法
  - 根據「閾值」(類似臨界值) 進行轉換
  - 某個像素的灰度值大於閾值，則轉換為黑色
  - 某個像素的灰度值小於閾值則轉換為白色
- 許多影像辨識或影像處理的領域 (例如輪廓偵測、邊緣偵測... 等)，都會使用二值化影像進行運算



原影像　　二值化後影像

OpenCV影像二值化

- cv2.threshold(img_gray, thresh, maxval, type)
  - 將灰階的影像，以二值化的方式轉換成黑白影像
  - img 影像物件
  - thresh 閾值，通常設定 127
  - maxval 最大灰度，通常設定 255
  - type 轉換方式
- threshold()會回傳兩個值
  - 第一個值為是否轉換成功(bool)
  - 第二個值為轉換後的輸出影像

OpenCV影像二值化

› 轉換方式有以下幾種

轉換方式	說明
cv2.THRESH_BINARY	如果大於 127 就等於 255，反之等於 0。
cv2.THRESH_BINARY_INV	如果大於 127 就等於 0，反之等於 255。
cv2.THRESH_TRUNC	如果大於 127 就等於 127，反之數值不變。
cv2.THRESH_TOZERO	如果大於 127 數值不變，反之數值等於 0。
cv2.THRESH_TOZERO_INV	如果大於 127 等於 0，反之數值不變。

OpenCV自適應二值化影像

- 自適應二值化
  - 根據指定大小的區域平均值
  - 整體影像的高斯平均值
  - 判斷所需的灰度和閾值
- threshold() 方法轉換灰階的影像時，必須手動設定灰度和閾值，比較適合內容較單純的影像
- adaptiveThreshold()適合處理比較複雜的影像，例如每個像素間可能都有關連性

OpenCV自適應二值化影像

- cv2.adaptiveThreshold(img, maxValue, adaptiveMethod, thresholdType, blockSize, C)
  - img 影像物件
  - maxValue 最大灰度，通常設定 255
  - adaptiveMethod 自適應二值化計算方法
  - thresholdType 二值化轉換方式
  - blockSize 轉換區域大小，通常設定 11
  - C 偏移量，通常設定 2

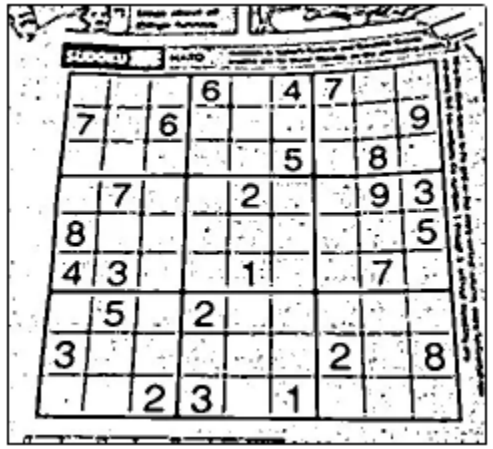
自適應二值化方法	說明
cv2.ADAPTIVE_THRESH_MEAN_C	使用區域平均值。
cv2.ADAPTIVE_THRESH_GAUSSIAN_C	使用整體高斯平均值。

OpenCV降低雜訊

- 使用模糊化功能先將影像模糊化
  - img_gray = cv2.medianBlur(img_gray, 5)
- 再將影像轉為二值化影像
  - output = cv2.adaptiveThreshold(img_gray, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY, 11, 2)

原圖

直接二值化



有模糊化再二值化

練習

- 讀取image資料夾內的20張image
- 將影像做以下操作(用multithread方法加速)
  - 第1-5張做雙邊模糊的模糊化
  - 第6-10張做先侵蝕後膨脹的降低雜訊
  - 第11-15張先將影像都轉成灰階然後做自適應二值化
  - 第16-20張先做中值濾波模糊再二值化的降低雜訊操作
- 儲存處理過後的影像