HW3\_影像還原 Adaptive Median Filter

1. 影像還原-適應性中間值濾波器

Adaptive Median Filter是一種用於去除雜訊的圖像處理演算法，通過在圖像上應用中值濾波器來實現去除雜訊的效果，與中值濾波器不同的是Adaptive Median Filter能夠根據圖像的不同區域適應性地調整濾波的大小，從而更加有效地去除雜訊。

Adaptive Median Filter的基本原理是，對於每個像素，它會先設置一個window，並計算出該區域內所有像素的中間值、最小值、最大值以及目前像素值，並透過第二點之演算法進行運算，最後得到去除雜訊後的圖片。

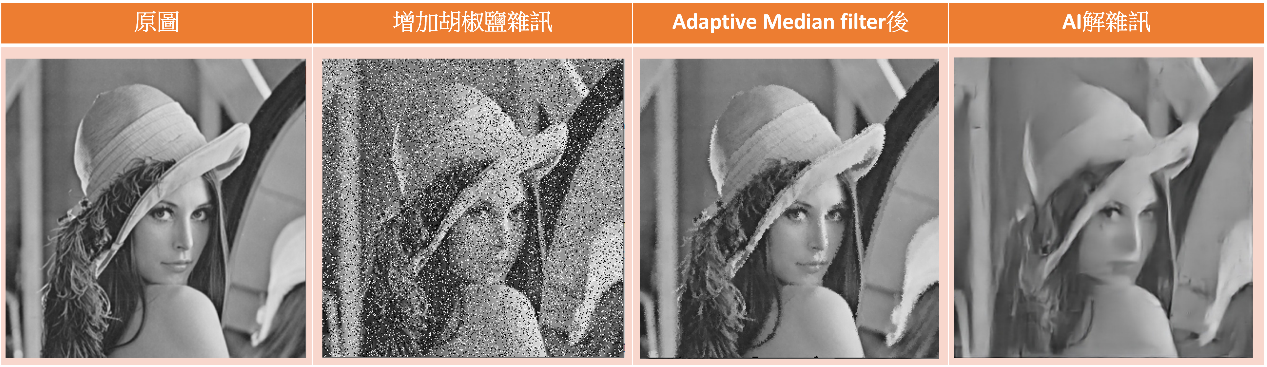


圖 一 影像還原過程

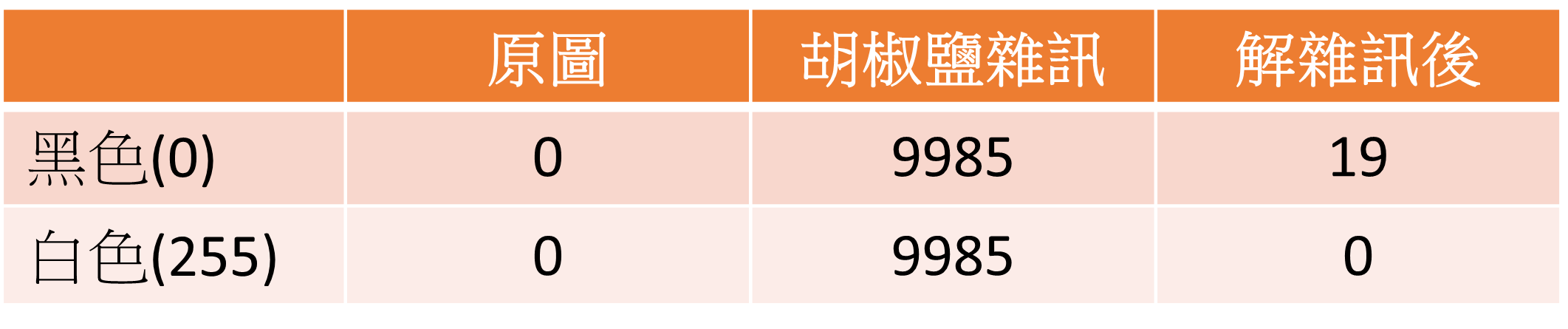


圖 二 雜訊總量

1. 演算法

符號如下：

演算法如下：

依照上方的演算法持續運算，就能夠有效地去除雜訊。

三、程式實作

首先，我們先透過cv2.imread讀取圖片(img)，因為Adaptive Median Filter只能在圖片為灰階時執行，因此我們透過trans\_gray將img轉為灰階色彩，並透過calculate\_noise函數記錄此時灰階圖片的0(黑色)與255(白色)，從程式的輸出結果來看，可以發現此時沒有像素等於0與255。



圖 三 讀取圖片、轉灰階及計算0與255的個數



圖 四 轉灰階函數

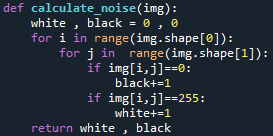


圖 五 計算0與255的函數內容



圖 六 運算結果

得到此時的0與255後，可以透過salt\_pepper\_noise函數將灰階圖片撒滿雜訊。

Salt\_pepper\_noise首先會先讀取圖片的row與col，之後建立白雜訊與黑雜訊的數量為 ，確保白雜訊與黑雜訊的數量為總圖片的20%且各半，之後進入while迴圈，當白雜訊或黑雜訊還有數量時，就會一直執行，進入後先使用random.random()建立一個機率值prob，prob會藉於0~1之間，再使用random.randint(0,row-1)及random.randint(0,col-1)讓程式幫我們生成要撒雜訊的位置，分別給noise\_row變數及noise\_col變數，若prob>0.5，就需要再確保黑雜訊還有數量(black\_noise>0)，通過後需要再確保此時的位置還未被撒過雜訊(使用if判斷該位置是否等於0或255，若相等，就跳脫此次的迴圈)，都通過後，即可將該點(noise\_row,noise\_col)設置為0，並且將黑雜訊數量-1(black\_noise -1)，直至黑雜訊與白雜訊的數量都為0後才結束迴圈，白雜訊邏輯與上述方式相同，不再贅述，這個函數可以確保我們只會撒20%的雜訊，並且黑雜訊與白雜訊各一半，也確保雜訊不會重複撒在同一個像素上。

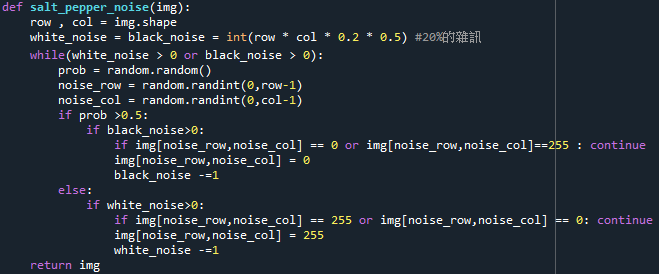


圖 七 salt\_pepper\_noise函數內容



圖 八 撒完雜訊後雜訊數量



圖 九 撒胡椒鹽雜訊後的結果

撒完雜訊後，我們就能來解雜訊了，首先我們將撒完雜訊後的圖片傳進Adaptive\_Median\_Filter函數內，函數內我們先定義Windwo\_size 為3，並且使用np.zeros\_like生成一張與原圖一樣大小的圖片(output)，為了避免我們的window超出圖片邊界，因此我們再使用np.pad將圖片做padding的動作，之後再使用for迴圈將整張圖片跑過一輪，每一個像素點我們都傳入Adaptive\_Median內進行計算。

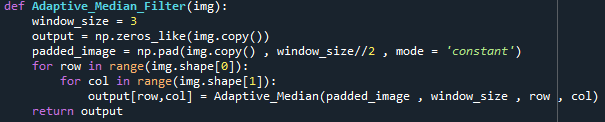


圖 十 Adaptive\_Median\_Filter函數內容

Adaptive\_Median函數會接收到該圖片、window\_size以及row、col，此時我們先設定window\_size最大只能到7，之後設置一個變數為window，內容為原圖片的，即能夠計算在當前像素值加上window\_size後所有像素值的最小值、最大值、中值，但計算前我們需要先將剛剛設置的window使用flatten()展平再透過np.sort排序，才能進行計算。

取得該點的像素值、window內的最大值、最小值及中值後，即可進入我們的演算法內計算，我們先判斷是否，再判斷是否，若都成立，即可將當前值回傳，並填入我們剛剛所建立的output內，若 成立但 不成立，則回傳中間值，若一開始的 就不成立了的話，就會將window\_size增加2，再判斷window\_size是否小於我們初始建立的7，是的話我們就能透過遞迴的方式將新的window\_size傳給Adaptive\_Median，若超過7的話，就直接回傳中間值，至此完成我們的演算法，也一點一點地將值填入output中，當所有像素值跑完也等同於完成解雜訊了。

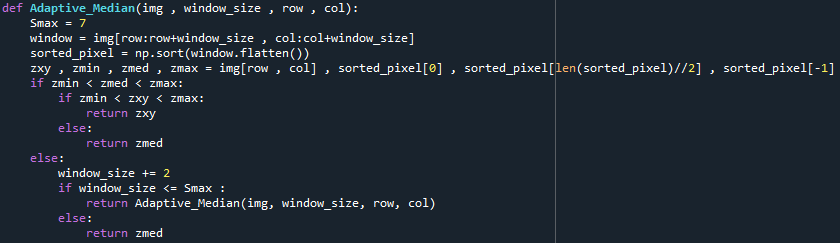


圖 十一 Adaptive\_Median函數



圖 十二 解雜訊後的雜訊數量



圖 十三 解雜訊後的結果

之後我們嘗試使用AI的方式替我們解決胡椒鹽雜訊，我將撒完胡椒鹽雜訊後的結果上傳至[AI.IMAGE ENLARGER]( <https://imglarger.com/Denoiser>)，此網站標榜AI Denoiser,Clean Your Photo，但回傳的結果如圖十四，看起來差強人意了點。



圖 十四 AI解雜訊

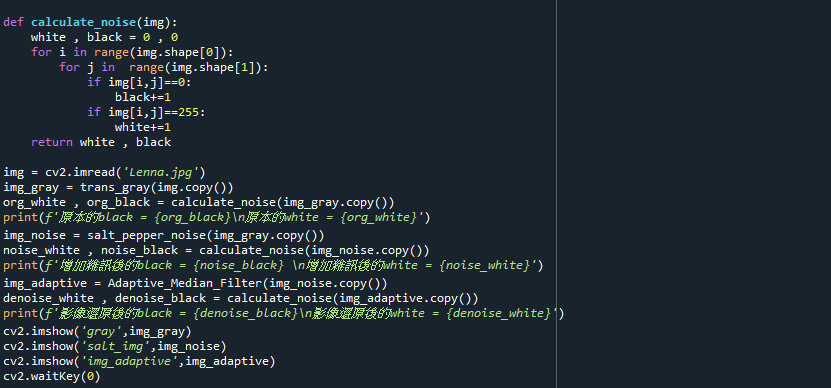
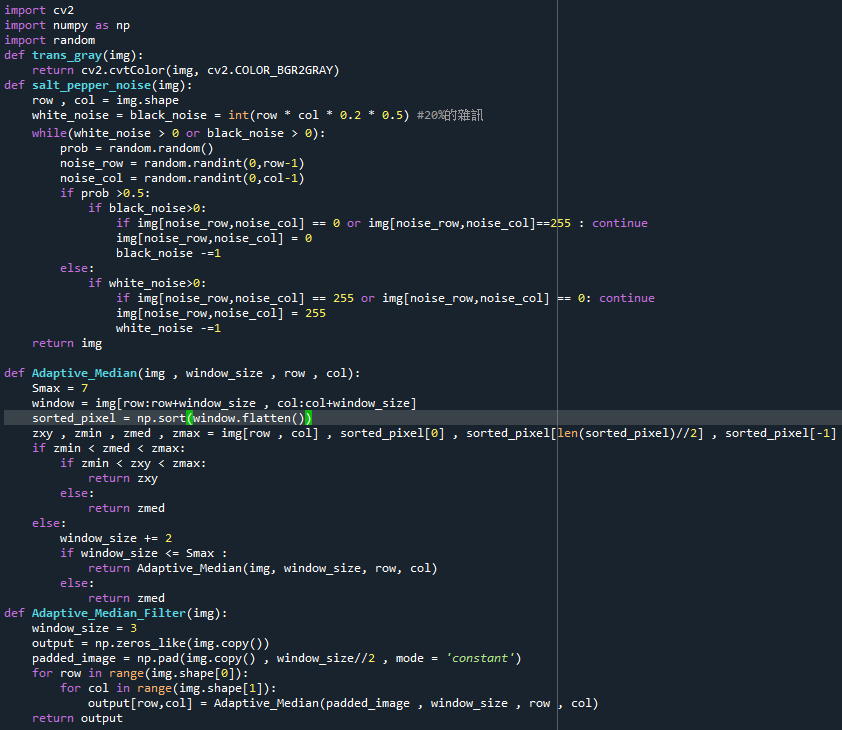
完整程式碼如下：

圖 十五 程式碼



四、心得

此次作業其實不難，只需要照著老師上課所說的演算法實際順過一輪，就可以寫出來，只是前面需要先思考如何撒雜訊才能有效地只撒20%以及黑白雜訊要平均，幾經思考過後才寫出此版本，雖然使用while迴圈較複雜，但總體來說能公平有效地撒雜訊才是我們目前所需要的。

老師上課有提到希望我們將自己解的雜訊與AI解的雜訊做比較，在我所找到的AI解雜訊網站可能較不適合解胡椒鹽雜訊，所以才會有此結果，但我相信在其他雜訊的方面AI一定會做的更好。