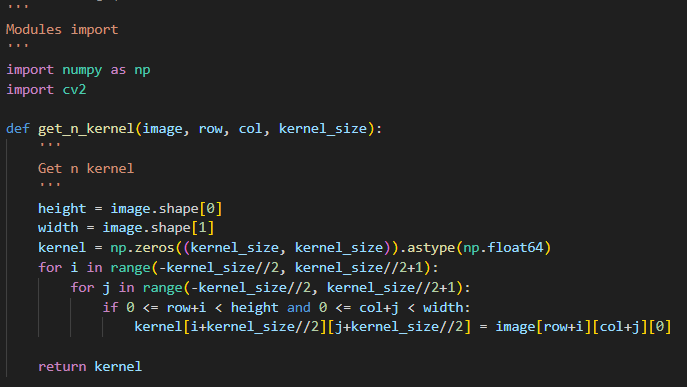
班級：資工三

學號：110590034

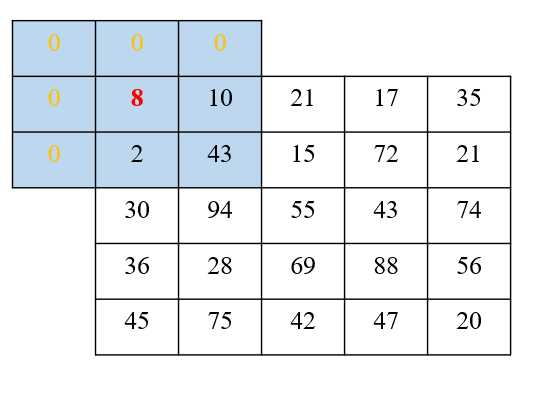
姓名：楊榮鈞

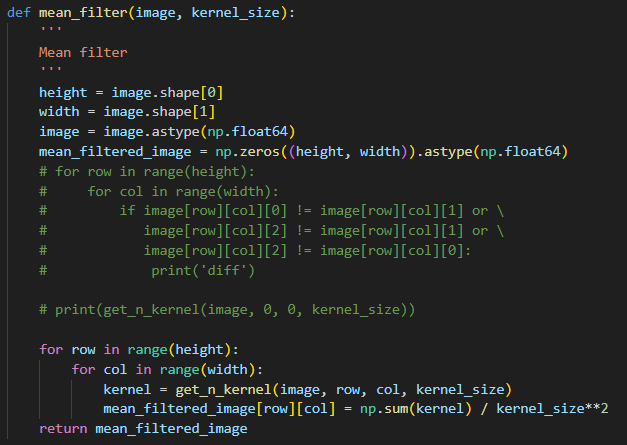


Function get\_n\_kernel:

先用image.shape找出height和width，再根據row和col(當前pixel的位置)抓取kernel\_size x kernel\_size大小的kernel。

如作業附圖的藍色的部分。



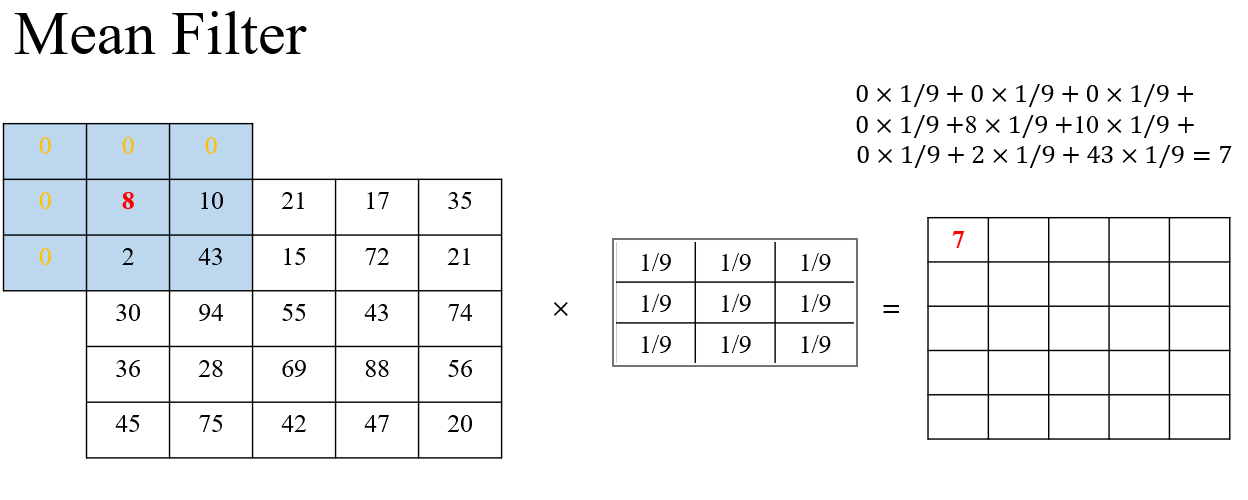


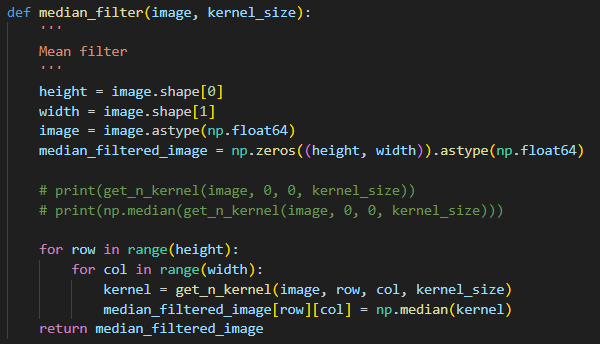
Function mean\_filter:

利用shape找出image的height和width，並利用astype設定image的type為np.float64，然後利用np.zeros創建mean\_filtered\_image的array。

註解的部分是測試image的三個通道的value是否皆相等，由於之前測試的結果皆為相等，因此有先註解使程式跑快一點。

接著利用for迴圈將計算每個pixel的value，其中會先用get\_n\_kernel抓取當前pixel的kernel，並利用np.sum(kernel) / kernel\_size \*\* 2計算mean filter，如同作業的PPT提到的算法，我將1/9的部分(1/ kernel\_size \*\* 2)提出來在最後相乘。

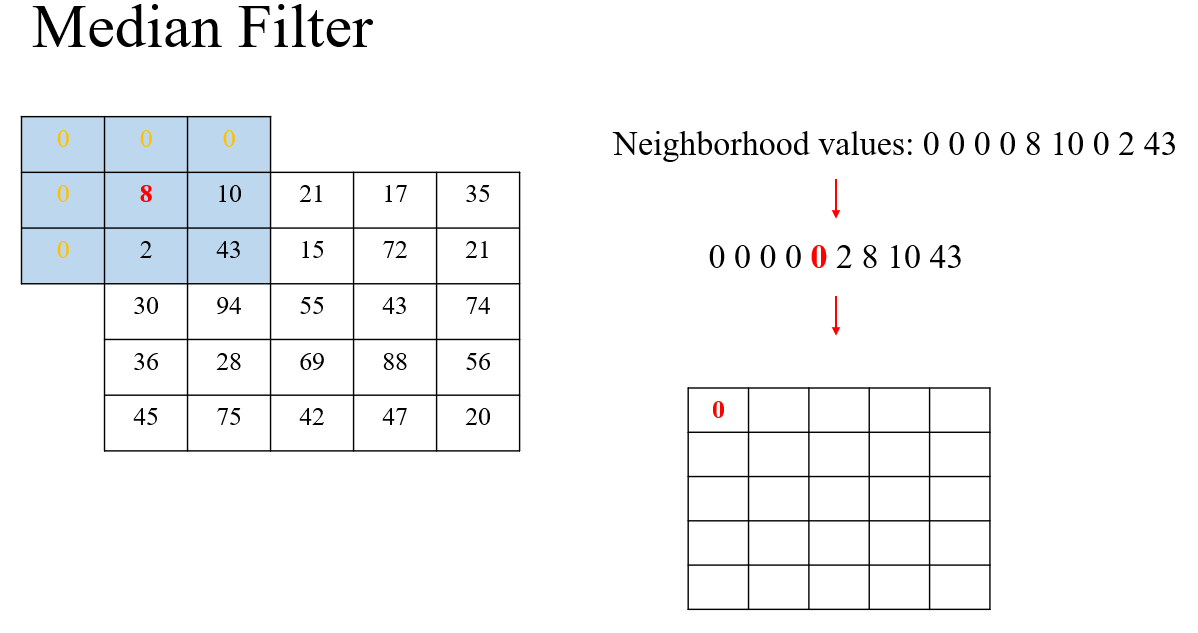


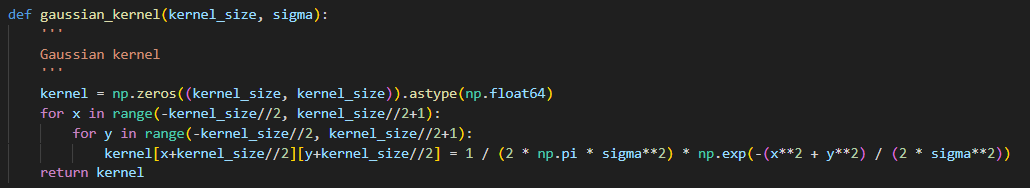


Function median\_filter:

利用shape找出image的height和width，並利用astype設定image的type為np.float64，然後利用np.zeros創建median\_filtered\_image的array。

接著利用for迴圈將計算每個pixel的value，其中會先用get\_n\_kernel抓取當前pixel的kernel，並利用np.median(kernel)找出kernel中的中值(median value)。

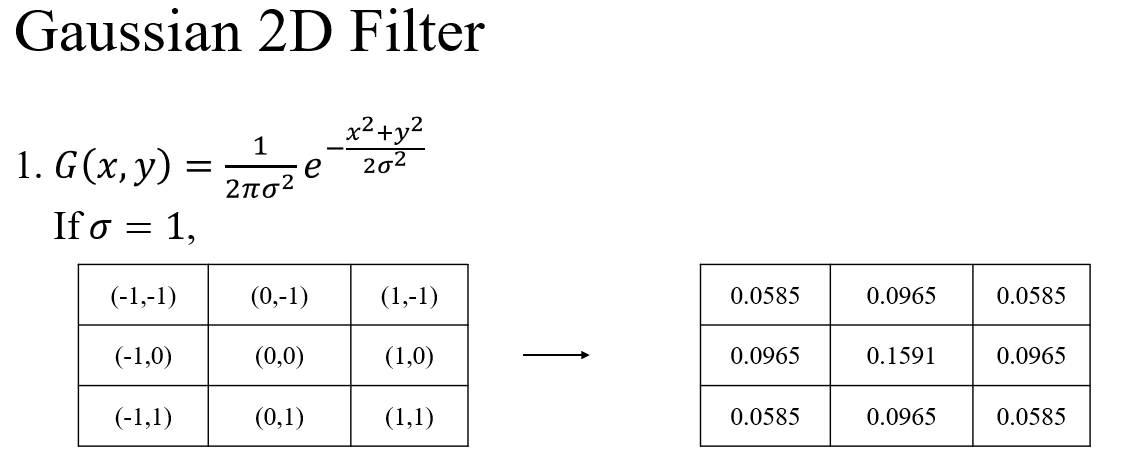


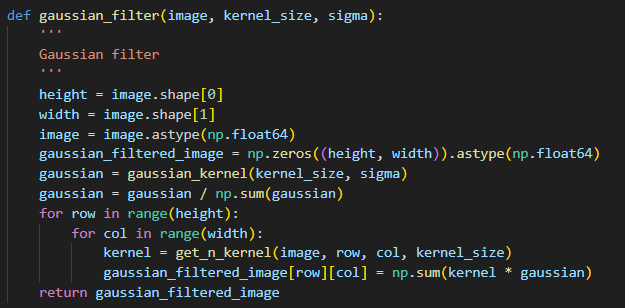


Function gaussian\_kernel:

計算gaussian的value，以Gaussian 2D filter的公式進行計算，如作業簡報的公式算法 G(x, y) = (1 / 2 \* pi \* sigma \*\* 2) \* exp \*\* (-(x \*\* 2 + y \*\* 2) / (2 \* sigma \*\* 2))。

其中會以kernel\_size的大小去做整個kernel的處理如圖所示(圖片是3x3的kernel大小)。



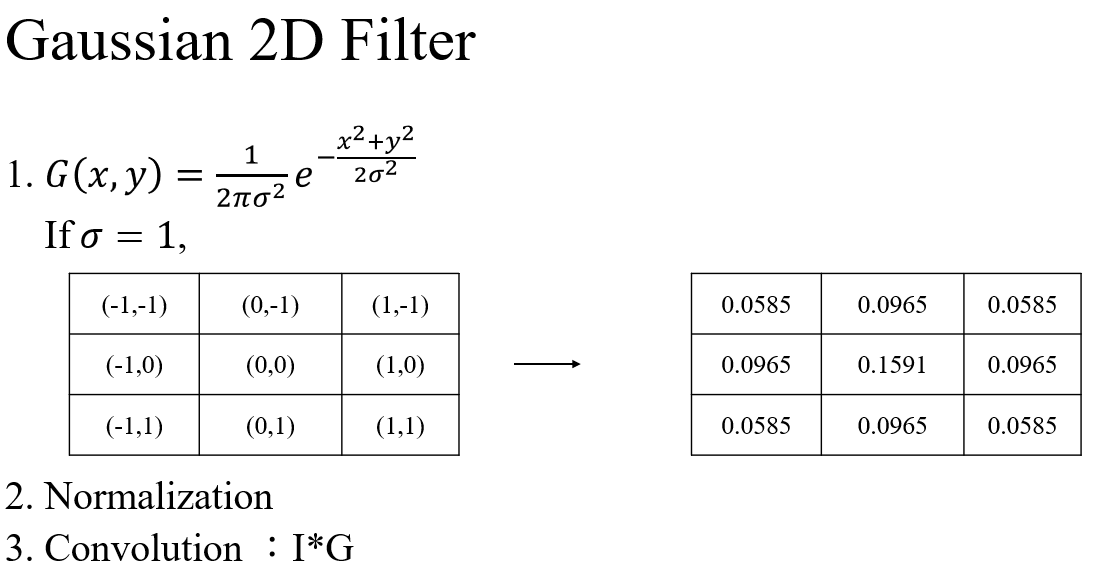


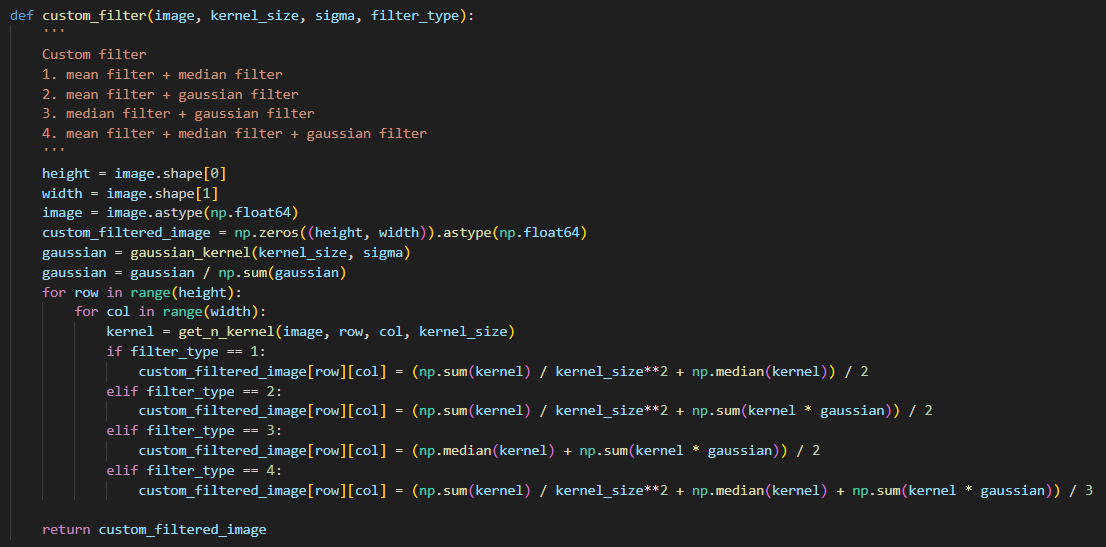
Function gaussian\_filter:

利用shape找出image的height和width，並利用astype設定image的type為np.float64，然後利用np.zeros創建gaussian\_filtered\_image的array。

利用gaussian\_kernel找出kernel的gaussian value，然後將gaussian除gaussian的總和並存在gaussian中。

利用for迴圈計算每個pixel的value，其中會先用get\_n\_kernel抓取當前pixel的kernel，並利用np.sum(kernel \* gaussian)得出kernel \* gaussian的總和，再存入當前pixel的value中。





Function custom\_filter:

結合前面的所提到的filter，組合成新的filter，其中有：(mean\_filter + median\_filter) / 2、(mean\_filter + gaussian\_filter) / 2、(median\_filter + gaussian\_filter) / 2和(mean\_filter + median\_filter + gaussian\_filter) / 3。

利用shape找出image的height和width，並利用astype設定image的type為np.float64，然後利用np.zeros創建custom\_filtered\_image的array。

利用gaussian\_kernel找出kernel的gaussian value，然後將gaussian / gaussian的總和並存在gaussian中，利用for迴圈計算每個pixel的value，其中會先用get\_n\_kernel抓取當前pixel的kernel，並根據filter\_type決定接下來的步驟。

當filter\_type == 1時，會計算(mean\_filter + median\_filter) / 2，也就是(np.sum(kernel) / kernel\_size\*\*2 + np.median(kernel)) / 2，並把結果的value存入custom\_filtered\_image當前的pixel的value中。

當filter\_type == 2時，會計算(mean\_filter + gaussian\_filter) / 2，也就是(np.sum(kernel) / kernel\_size\*\*2 + np.sum(kernel \* gaussian)) / 2，並把結果的value存入custom\_filtered\_image當前的pixel的value中。

當filter\_type == 3時，會計算(median\_filter + gaussian\_filter) / 2，也就是(np.median(kernel) + np.sum(kernel \* gaussian)) / 2，並把結果的value存入custom\_filtered\_image當前的pixel的value中。

當filter\_type == 4時，會計算(mean\_filter + median\_filter + gaussian\_filter) / 3，也就是(np.sum(kernel) / kernel\_size\*\*2 + np.median(kernel) + np.sum(kernel \* gaussian)) / 3，並把結果的value存入custom\_filtered\_image當前的pixel的value中。



Function image:

先用cv2的imread讀取圖片到origin\_image。

接著使用mean\_filter(origin\_image, 3)得出mean\_filter且kernel為3x3大小的圖片並存入mean\_filtered\_image\_3中，再來使用cv2.imwrite存檔，最後再使用cv2.imread讀取圖片到mean\_3。

接著使用mean\_filter(origin\_image, 7)得出mean\_filter且kernel為7x7大小的圖片並存入mean\_filtered\_image\_7中，再來使用cv2.imwrite存檔，最後再使用cv2.imread讀取圖片到mean\_7。

接著使用median\_filter(origin\_image, 3)得出median\_filter且kernel為3x3大小的圖片並存入median\_filtered\_image\_3中，再來使用cv2.imwrite存檔，最後再使用cv2.imread讀取圖片到median\_3。

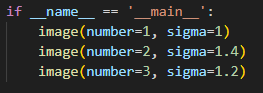
接著使用median\_filter(origin\_image, 7)得出median\_filter且kernel為7x7大小的圖片並存入median\_filtered\_image\_7中，再來使用cv2.imwrite存檔，最後再使用cv2.imread讀取圖片到median\_7。

接著使用gaussian\_filter(origin\_image, 5, sigma)得出gaussian\_filter且kernel為5x5大小和傳入的sigma做運算後的圖片，並存入gaussian\_filtered\_image中，再來使用cv2.imwrite存檔，最後再使用cv2.imread讀取圖片到gaussian。

然後利用for迴圈得出custom\_filter的4種filter type的圖片，並使用cv2.imwrite存檔。

利用cv2的imshow查看original image、mean\_3(Mean Filtered Image (3x3))、mean\_7(Mean Filtered Image (7x7))、median\_3(Median Filtered Image (3x3))、median\_7(Median Filtered Image (7x7))和gaussian(Gaussian Filtered Image)。

利用cv2的waitKey(0)和destroyAllWindows()將imshow顯示出來的圖片關閉。



執行image，產生所有助教給的圖片的Mean Filtered Image (3x3 kernel size)、Mean Filtered Image (7x7 kernel size)、Median Filtered Image (3x3 kernel size)、Median Filtered Image (7x7 kernel size)、Gaussian Filtered Image以及自己製作的filter image。

Result images

img1\_q1\_3.jpg



我預期的結果與輸出大致相同，與原圖相比模糊程度輕微，細節略有損失。

img1\_q1\_7.jpg



我預期的結果與輸出大致相同，與原圖相比模糊程度輕微，細節略有損失。

使用mean\_filter的結果看起來是比較像是模糊化影像，其中7x7大小的kernel模糊的程度比3x3高。

img1\_q2\_3.jpg



我預期的結果與輸出大致相同，與原圖相比去除了部分noise value，圖像相對清晰。

img1\_q2\_7.jpg



我預期的結果與輸出大致相同，與原圖相比去除了更多noise value，但同時細節也有所減少，整體效果比3x3 kernel更為平滑。

使用median\_filter的結果看起來會filter掉一些noise value，像是樹枝上的斑點，從7x7大小的kernel看，會發現幾乎消失掉，其中7x7大小的kernel的filter程度比3x3高。

img1\_q3.jpg



我預期的結果與輸出大致相同，與原圖相比gaussian\_filter的效果介於mean\_filter和median\_filter之間，模糊效果比較均勻，細節保持相對較好。

img2\_q1\_3.jpg



我預期的結果與輸出大致相同，與原圖相比模糊程度輕微，細節略有損失。

img2\_q1\_7.jpg



我預期的結果與輸出大致相同，與原圖相比模糊程度增加，細節損失更多，圖像變得更加平滑。

使用mean\_filter的結果看起來是比較像是模糊化影像(7x7 kernel有種144p的感受)，其中7x7大小的kernel模糊的程度比3x3高。

img2\_q2\_3.jpg



我預期的結果與輸出大致相同，與原圖相比減少了圖像中的noise value，保持了較多的細節。

img2\_q2\_7.jpg



我預期的結果與輸出大致相同，與原圖相比更多noise value被去除，但圖像細節也有所損失，整體效果比3x3 kernel更為平滑。

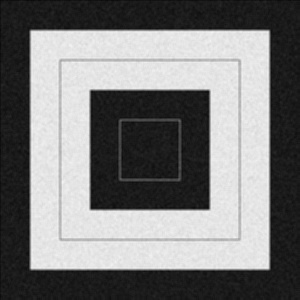
使用median\_filter的結果會filter掉一些noise value，像是圖片上的一堆noise value，其中7x7大小的kernel的filter程度比3x3高，但是7x7的kernel也會讓老虎的細節損失更多。

img2\_q3.jpg



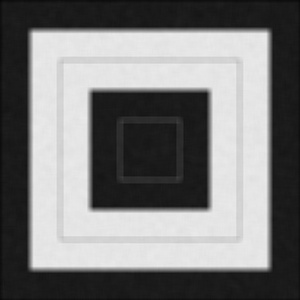
我預期的結果與輸出大致相同，與原圖相比gaussian\_filter對圖像進行了均勻的模糊處理，細節保持較好，圖像變得更柔和。

img3\_q1\_3.jpg



我預期的結果與輸出大致相同，與原圖相比邊緣變得稍微模糊，圖像整體效果略有改變。

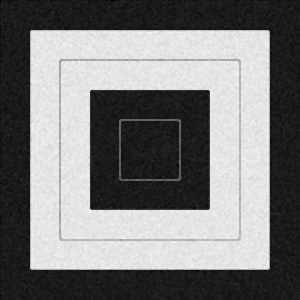
img3\_q1\_7.jpg



我預期的結果與輸出大致相同，與原圖相比邊緣模糊更加明顯，圖像變得更平滑。

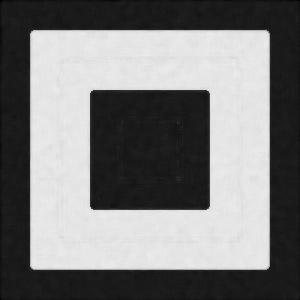
使用mean\_filter的結果看起來是模糊化影像，其中7x7大小的kernel模糊的程度比3x3高，而7x7大小的kernel的filter結果讓眼睛相對於3x3大小的kernel舒服許多。

img3\_q2\_3.jpg



我預期的結果與輸出大致相同，與原圖相比去除了部分noise value，邊緣保持相對清晰。

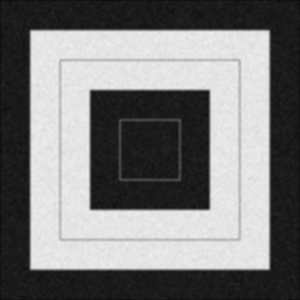
img3\_q2\_7.jpg



我預期的結果與輸出大致相同，與原圖相比去除了更多noise value，但邊緣模糊(白色方塊的黑色線條幾乎快看不到，只剩下隱約的線條，黑色方塊的白色線條也是相同結果)，圖像整體變得平滑。

使用median\_filter的結果會filter掉一些noise value，其中7x7大小的kernel的filter程度比3x3高，但是7x7的kernel讓黑色線條(方形框的線條)和白色線條(方形框的線條)幾乎看不見。

img3\_q3.jpg



我預期的結果與輸出大致相同，與原圖相比gaussian\_filter對圖像進行了均勻的模糊處理，邊緣變得柔和，圖像看起來更平滑且細節保持較好。

Mean filter：適合進行輕度模糊處理，kernel 越大模糊效果越明顯，但會損失更多細節。

Median filter：適合filter noise value，保持邊緣清晰度，kernel 越大filter的noise value越多，但也會造成細節損失。

Gaussian filter：模糊處理效果均勻，能夠有效平滑圖像，同時保持較好的細節。

Extra result (custom filter, 5x5 kernel)

(mean\_filter + median\_filter) / 2

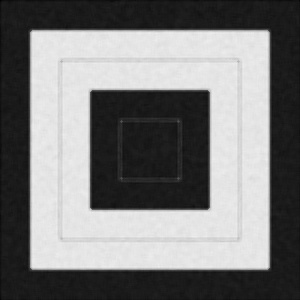
img1\_q4\_1.jpg



img2\_q4\_1.jpg



img3\_q4\_1.jpg



第一種方法我覺得效果不錯，有著mean\_filter和median\_filter的優點結合，其中有一些noise value有被filter掉，且細節沒有損失太多。

(mean\_filter + gaussian\_filter) / 2

img1\_q4\_2.jpg



img2\_q4\_2.jpg



img3\_q4\_2.jpg



第二種的方法相對於第一種，保留的細節較多，但老虎的圖片中，仍然有許多noise value。

(median\_filter + gaussian\_filter) / 2

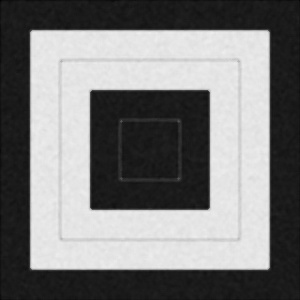
img1\_q4\_3.jpg



img2\_q4\_3.jpg



img3\_q4\_3.jpg



第三種方法相較於第一種，我覺得其保留的細節比較多一點，且noise value也有被filter一些。

(mean\_filter + median\_filter + gaussian\_filter) / 3

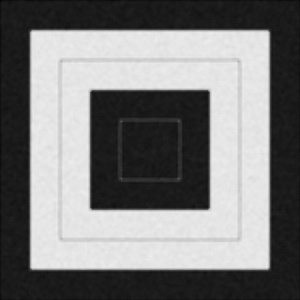
img1\_q4\_4.jpg



img2\_q4\_4.jpg



img3\_q4\_4.jpg



第四種方法，我覺得與第三種的觀感肉眼看起來差不多，只是noise value似乎比較多一點。

第一種方法：有filter掉noise value，但細節也有所損失。

第二種方法：保留較多細節，但留的noise value較多。

第三種方法：有filter掉noise value，細節相較於第一種方法損失較少。

第四種方法：與第三種方法相似，從老虎的圖片來看，似乎noise value被filter掉的程度比第三種方法低。

最後的結果總結：

這次hw5的作業難度在於理解三種filter的概念是什麼，並理解公式是如何運作，我覺得根據講義及作業的簡報內容來看，這次的難度沒有這麼高。其中經過這次作業我了解到：Mean filter適合輕度模糊處理，kernel越大模糊效果越明顯，但會損失更多細節；Median filter適合filter noise value，保持邊緣清晰度，kernel 越大filter的noise value越多，但也會造成細節損失；Gaussian filter模糊處理效果均勻，能夠有效平滑圖像，同時保持較好的細節。

其中我根據上面三種組合中我發現Median filter + Gaussian filter的filter效果還不錯，如果希望保存多一點細節的話可以用Mean filter + Median filter + Gaussian filter，若是希望可以去除較多noise value，且比較不在意一些細節損失的話，用Mean filter + Median filter效果會比較好，只是想要保留更多細節，而不在意noise value則是用Mean filter + Gaussian filter。