

# 1 a : 兩張圖片分別進行高斯模糊後的結果

由左至右依序為原圖，標準差為1的高斯模糊，標準差為5的高斯模糊

普遍認為，高斯模糊就是相機失焦時所得的影像，而「標準差」越大，看起來就像失焦得越嚴重的相片



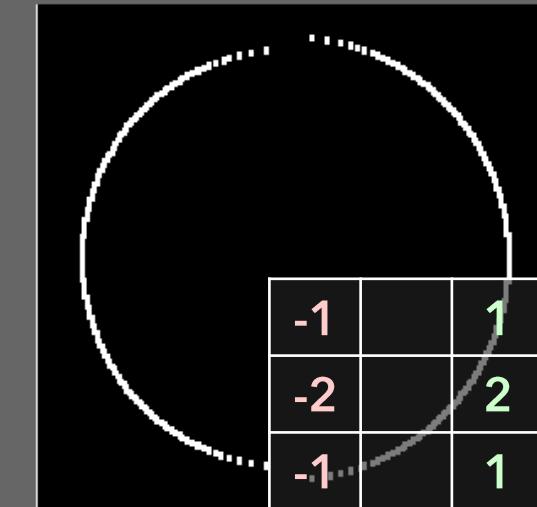
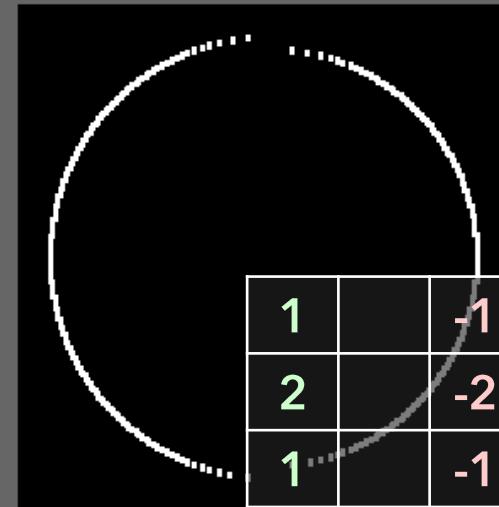
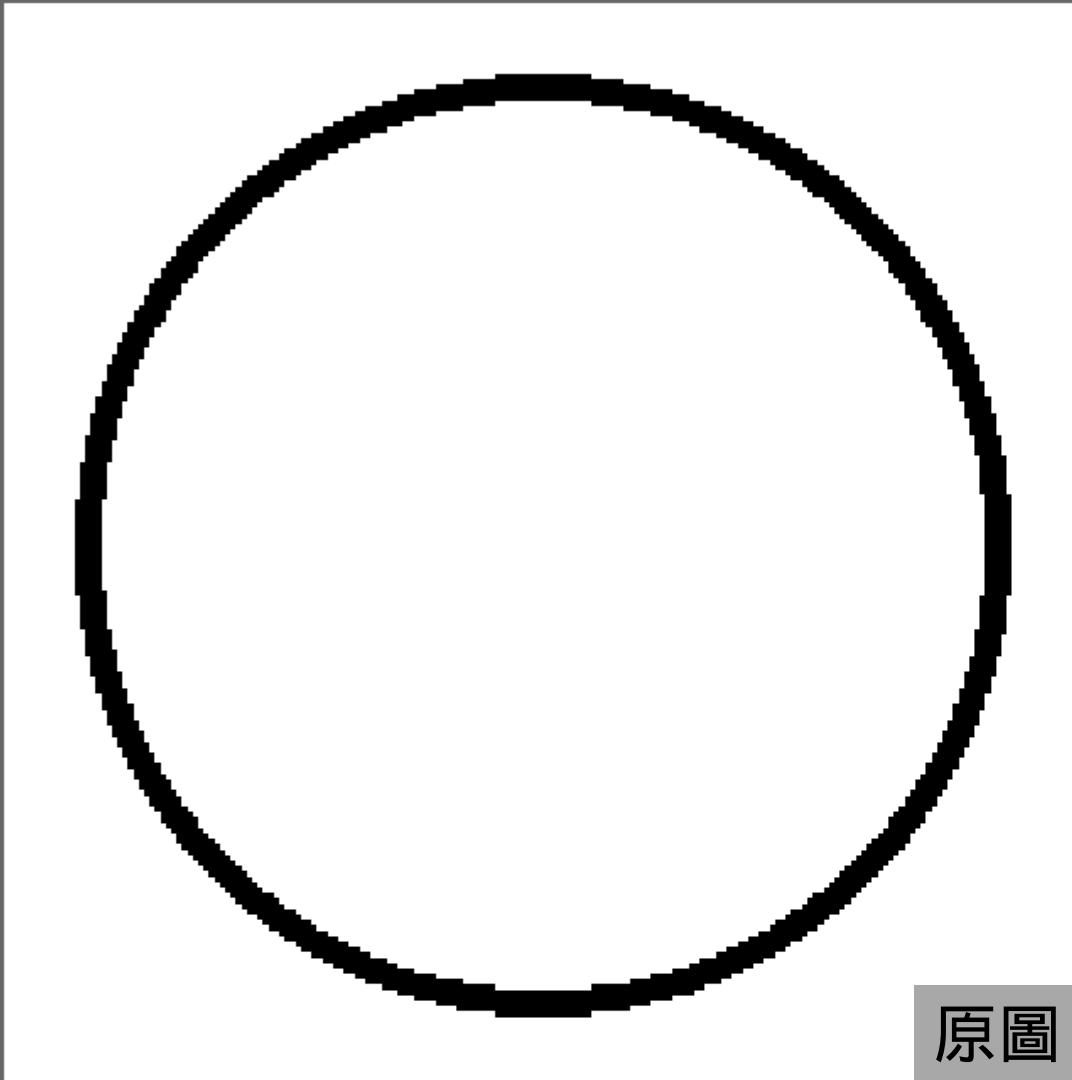
1 a : 透過對邊緣和角落進行特別處理(亮度補償, 如下列), 就不會導致黑邊(如上列)

由左至右依序為原圖, 標準差為1的高斯模糊, 標準差為5的高斯模糊

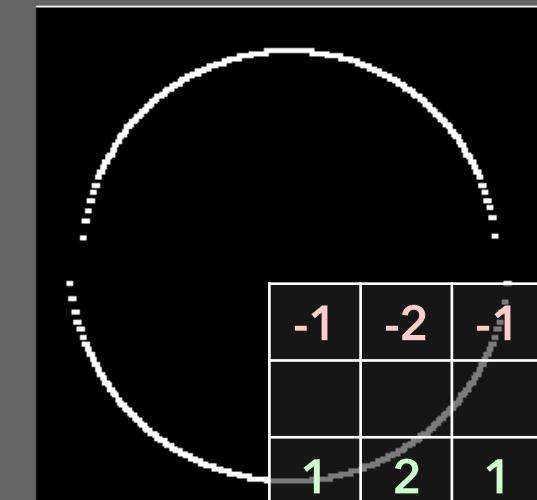
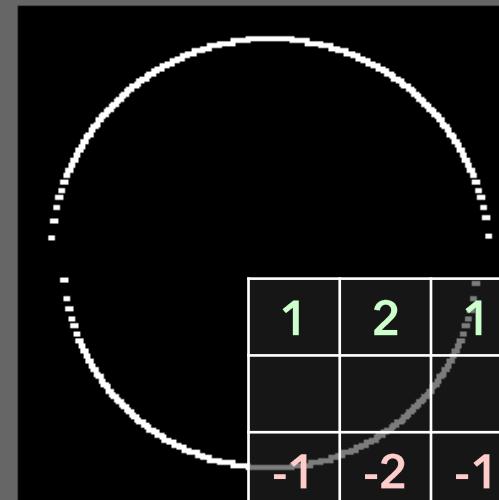


一張影像的四個角落和四個邊緣，  
是遮罩有可能「遮到外面」之處，  
必須對這些地方進行亮度補償

# 1 b : 4個方向的Sobel filter反應 (4個方向的Sobel filter的視覺化示例)



以左上圖為例，「左亮右暗」的情形大量出現在左半圓周的外側以及右半圓周的內側



# 1 b : 1 a 的4張圖片分別進行Sobel Filtering的結果

由左至右依序為原圖， 1a的4張圖經過Sobel Filter的結果

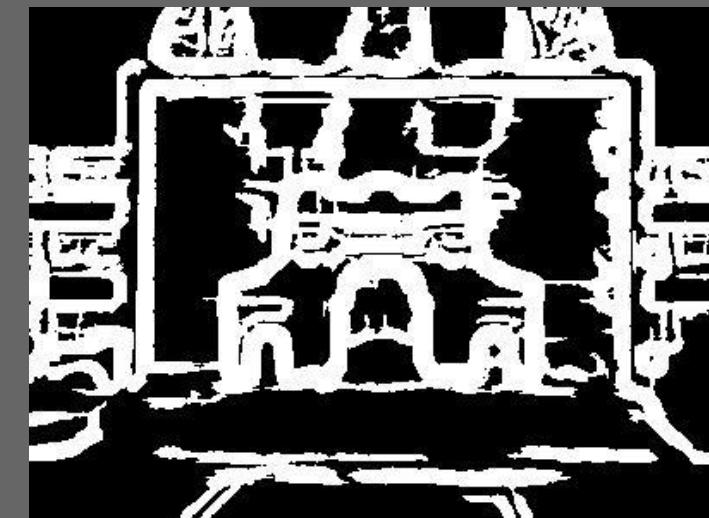
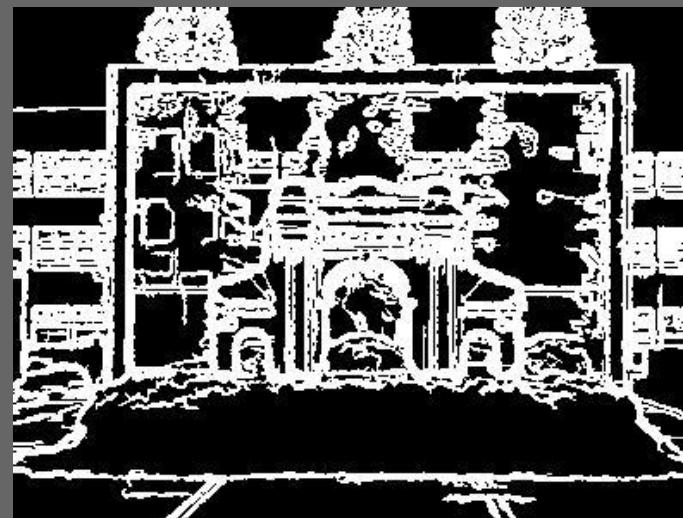
一樣地，在邊緣之處Sobel Filter可能會遮到外面，所以我將邊緣處像素值設為「靠內的鄰居」的像素值  
您可以看見，在越「平坦」的高斯遮罩下，影像的Level of Detail越低，因此對於邊緣偵測的反應也越低



# 1 c : 1 b 的4張圖片分別進行Non-maximum suppression的結果

由左至右依序為原圖， 1b的4張圖經過Non-maximal suppression的結果

不同的Threshold會導致不同的結果，這裡呈現的是我認為比較合理的結果



# 1 c : Threshold的調整，對於結果的影響

以Olaf(雪寶)為例，有些邊緣(特別是前景和背景交界處)較不明顯，太大的Threshold可能會濾掉這些邊緣



# 1 C : Non-maximal suppression的實作細節

輸入：原影像

1

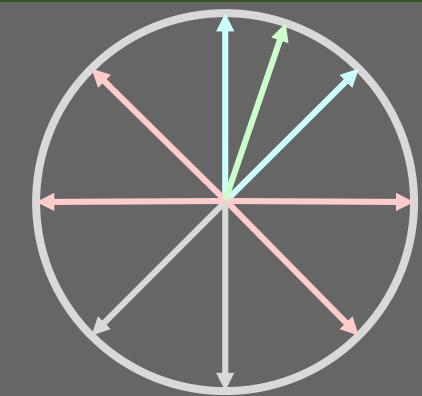
取得每個pixel的梯度方向和大小

2

取得每個pixel的切線方向  
如右圖

依照每個pixel的梯度方向

和8個基本方向做內積，  
取其最大(及次大)者，作為法線方向，  
垂直於法線方向的即為切線方向



3

取得區域極大值

該位置梯度大於Threshold1的，作為「種子」

從「種子」開始，進行BFS，往pixel的切線方向擴展，走到下一個pixel，

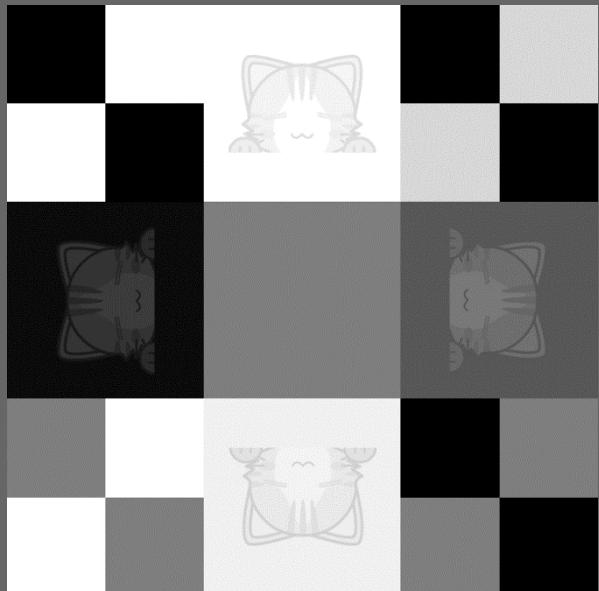
4

擴展邊緣(BFS)

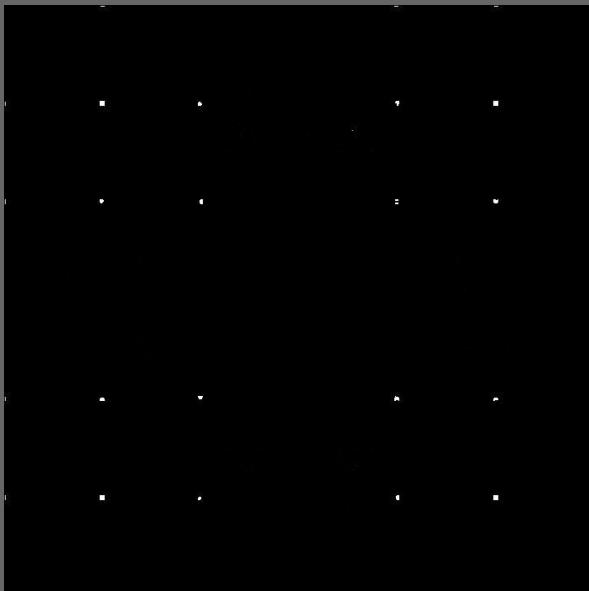
若它的梯度大小 $>$ Threshold2，則視其為邊緣，繼續往它的切線方向搜尋。  
若它的梯度大小 $<$ Threshold2，則這個方向的搜尋停止，  
繼續往其他可進行搜尋的方向搜尋邊緣，直到每個方向的擴展都到盡頭。

輸出：原影像經由non-maximal suppression所得到的邊緣

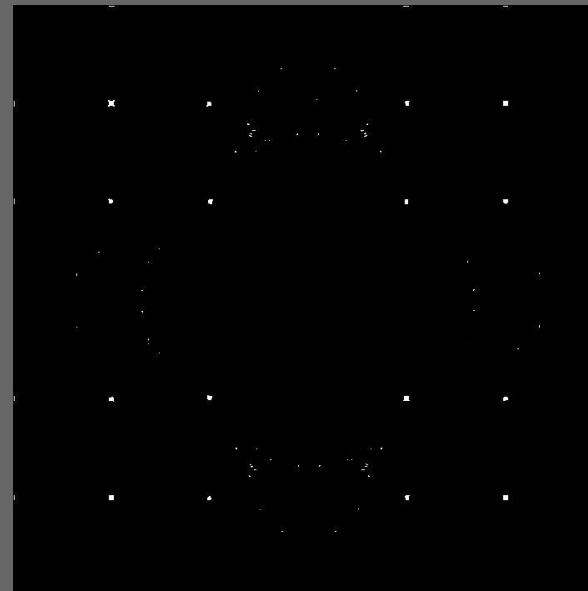
## 2 a 實驗 1/2：以特製的實驗用圖片，檢證演算法的正確性



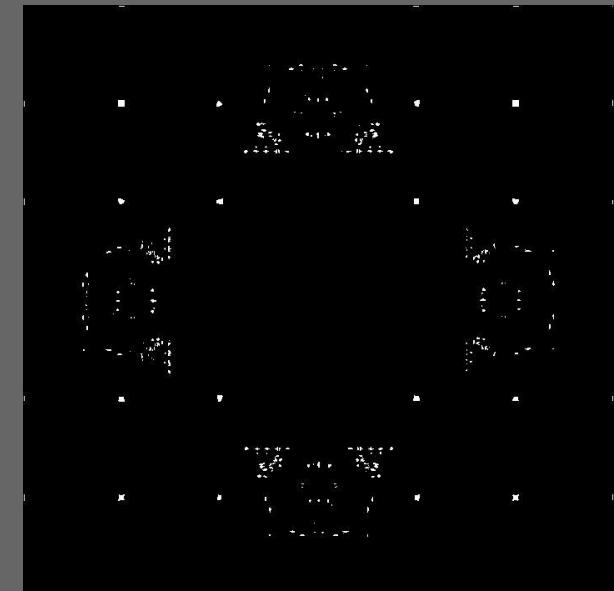
測試用棋盤圖片



經由大Threshold  
轉成黑白影像



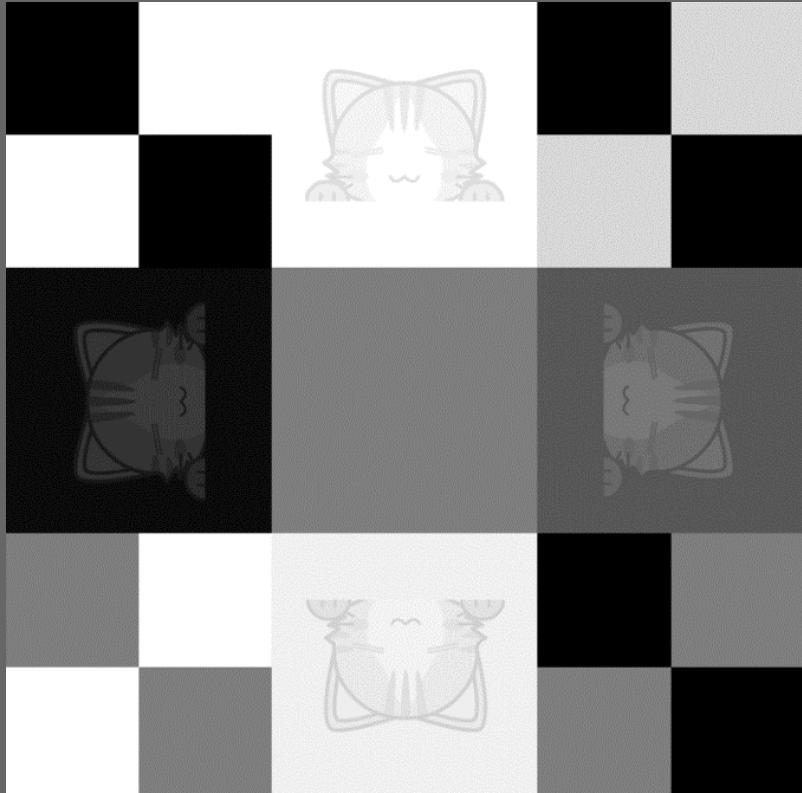
經由中Threshold  
轉成黑白影像



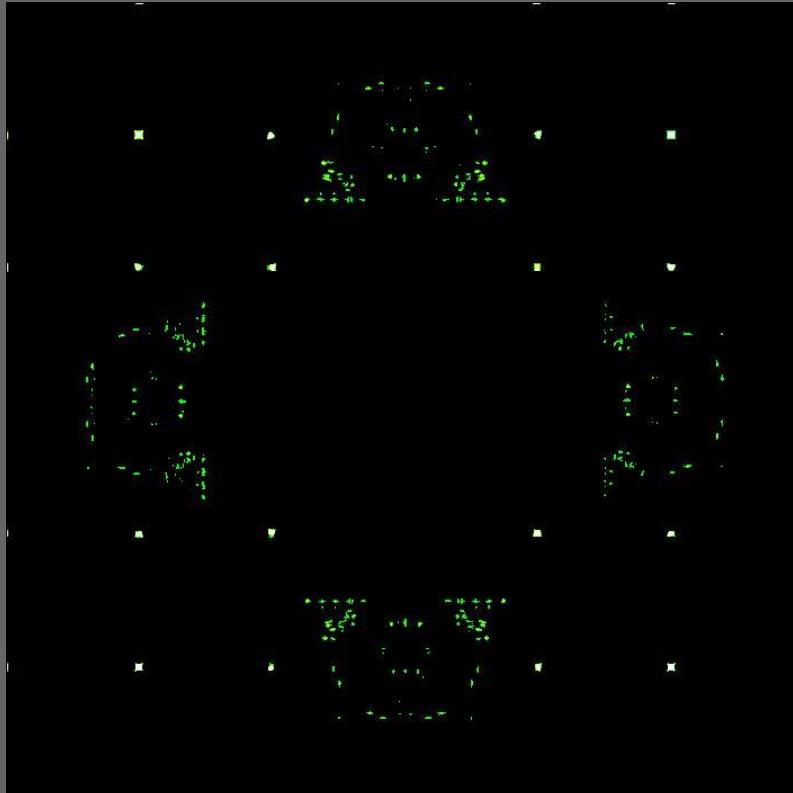
經由小Threshold  
轉成黑白影像

每個pixel的值，代表該處附近 $3 \times 3$ 的window，  
取其中的structure tensor( $2 \times 2$ )的eigenvalue中較小者

## 2 a 實驗 2/2：以特製的實驗用圖片，檢證演算法的正確性



測試用棋盤圖片



每個pixel的值，  
代表該處附近 $3 \times 3$ 的window，  
取其中的structure tensor( $2 \times 2$ )  
的eigenvalue中較小者

白色的點：該點值大於  
大Threshold

黃色的點：該點值大於  
中Threshold

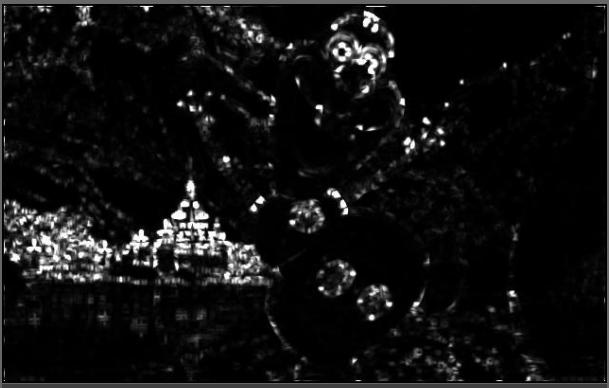
綠色的點：該點值大於  
小Threshold

## 2 a, 雪寶：Structure tensor中，較小的eigenvalue所形成的影像

每個pixel的值，代表該處附近 $3\times 3$ 或 $5\times 5$ 的window，取其中的structure tensor的eigenvalue中較小者



高斯模糊時 $\sigma=1$ ，  
Harris. window size =  $3\times 3$



高斯模糊時 $\sigma=1$ ，  
Harris. window size =  $5\times 5$



高斯模糊時 $\sigma=5$ ，  
Harris. window size =  $3\times 3$



高斯模糊時 $\sigma=5$ ，  
Harris. window size =  $5\times 5$



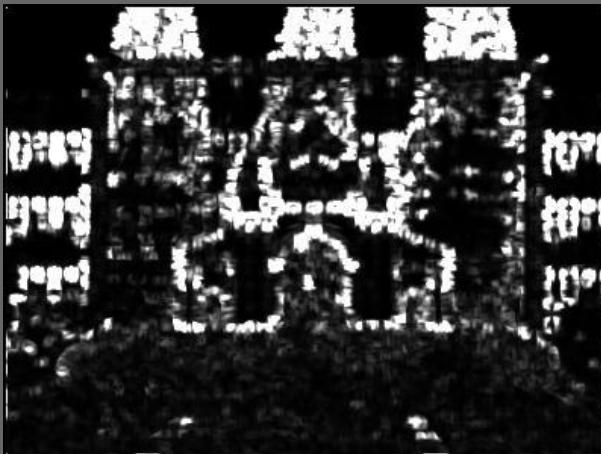
上圖以一個很小的，相同的Threshold轉成黑白圖片的結果

## 2 a, 清華園：Structure tensor中，較小的eigenvalue所形成的影像

每個pixel的值，代表該處附近 $3 \times 3$ 或 $5 \times 5$ 的window，取其中的structure tensor的eigenvalue中較小者



高斯模糊時 $\sigma=1$ ，  
Harris. window size =  $3 \times 3$



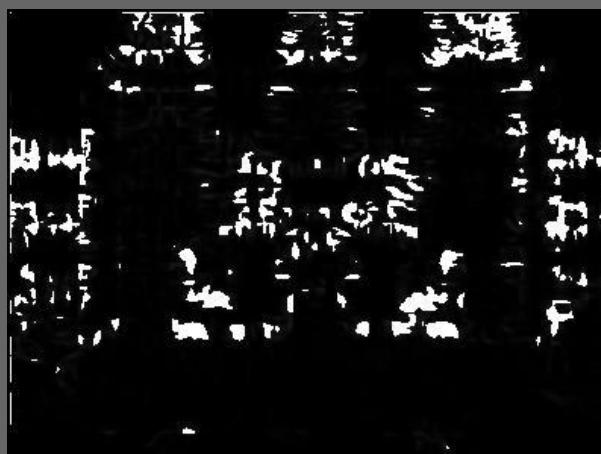
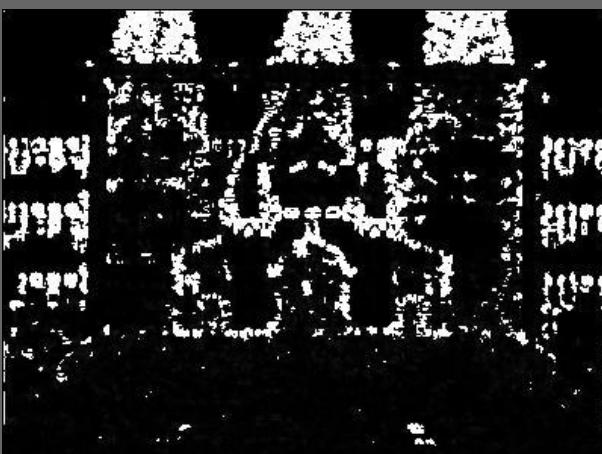
高斯模糊時 $\sigma=1$ ，  
Harris. window size =  $5 \times 5$



高斯模糊時 $\sigma=5$ ，  
Harris. window size =  $3 \times 3$

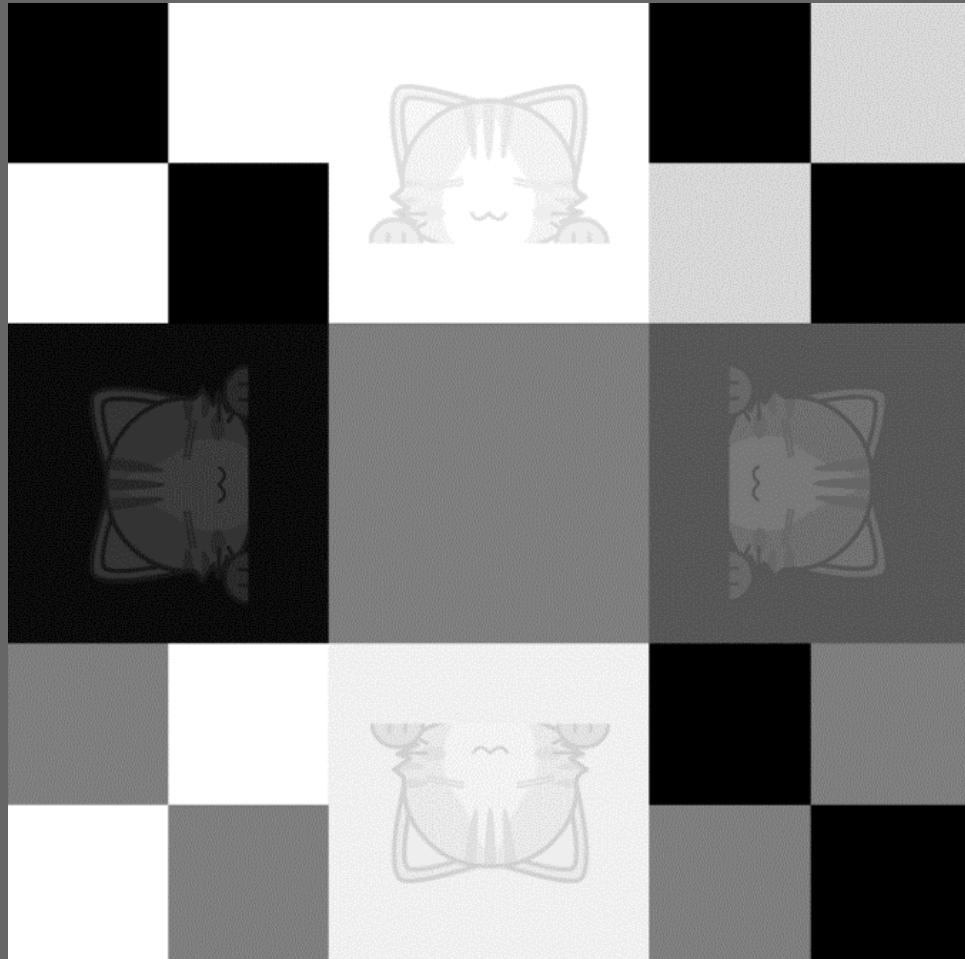


高斯模糊時 $\sigma=5$ ，  
Harris. window size =  $5 \times 5$

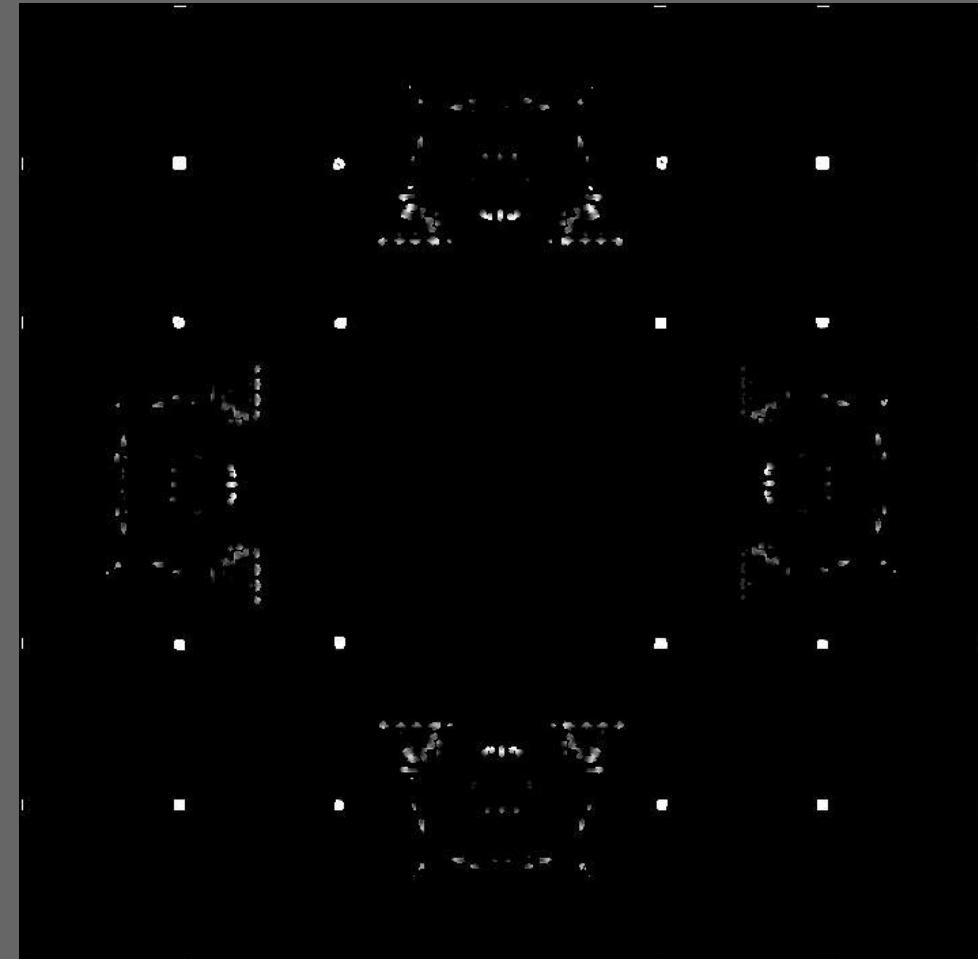


上圖以一個很小的，相同的Threshold轉成黑白圖片的結果

## 2 b 實驗 1/2：以特製的實驗用圖片，檢證演算法的正確性



測試用棋盤圖片



每個pixel的值，代表該處附近 $3 \times 3$ 的window的R，即corner response

## 2 b, 雪寶 : R (Corner Response) 所形成的影像

每個pixel的值，代表該處附近 $3 \times 3$ 或 $5 \times 5$ 的window，取其中的structure tensor的eigenvalue中較小者



高斯模糊時 $\sigma=1$ ，  
Harris. window size =  $3 \times 3$



高斯模糊時 $\sigma=1$ ，  
Harris. window size =  $5 \times 5$



高斯模糊時 $\sigma=5$ ，  
Harris. window size =  $3 \times 3$



高斯模糊時 $\sigma=5$ ，  
Harris. window size =  $5 \times 5$



上圖各自以一個適當的Threshold轉成黑白圖片的結果

## 2 b, 清華園 : R (Corner Response) 所形成的影像

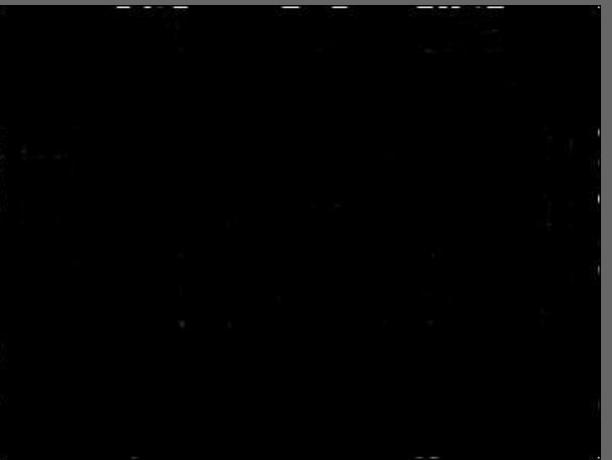
每個pixel的值，代表該處附近 $3 \times 3$ 或 $5 \times 5$ 的window，取其中的structure tensor的eigenvalue中較小者



高斯模糊時 $\sigma=1$ ，  
Harris. window size =  $3 \times 3$



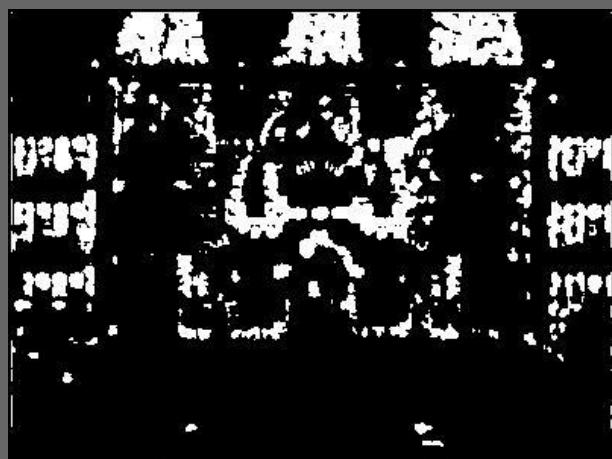
高斯模糊時 $\sigma=1$ ，  
Harris. window size =  $5 \times 5$



高斯模糊時 $\sigma=5$ ，  
Harris. window size =  $3 \times 3$



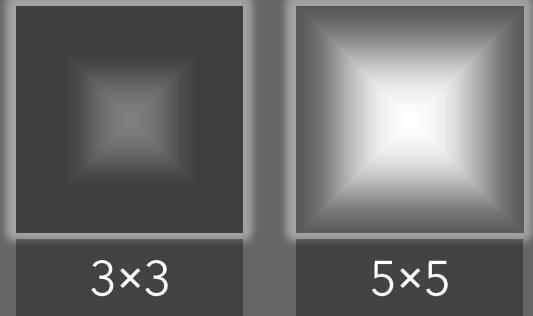
高斯模糊時 $\sigma=5$ ，  
Harris. window size =  $5 \times 5$



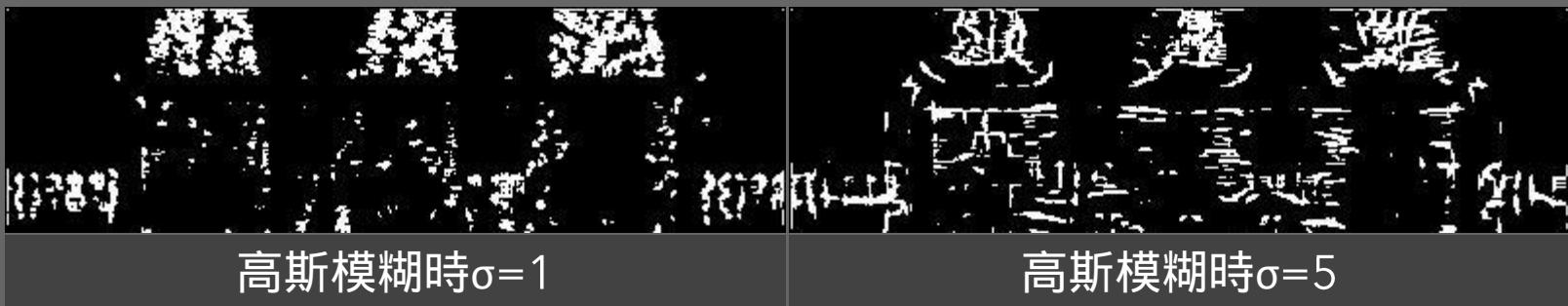
上圖各自以一個適當的Threshold轉成黑白圖片的結果

# 關於 2a 和 2b

1. 使用Window Size  $3 \times 3$  所找出來的角落，使用Window Size  $5 \times 5$  也能找到，看起來大概會像這樣：



2. 使用高斯模糊可以去雜訊，但原本解析度就不高的圖片，過度使用高斯模糊會將原有的不明顯的角落抹去，即使使用更低的threshold作為判斷是否為角落的標準，得到的結果品質也大不如前：

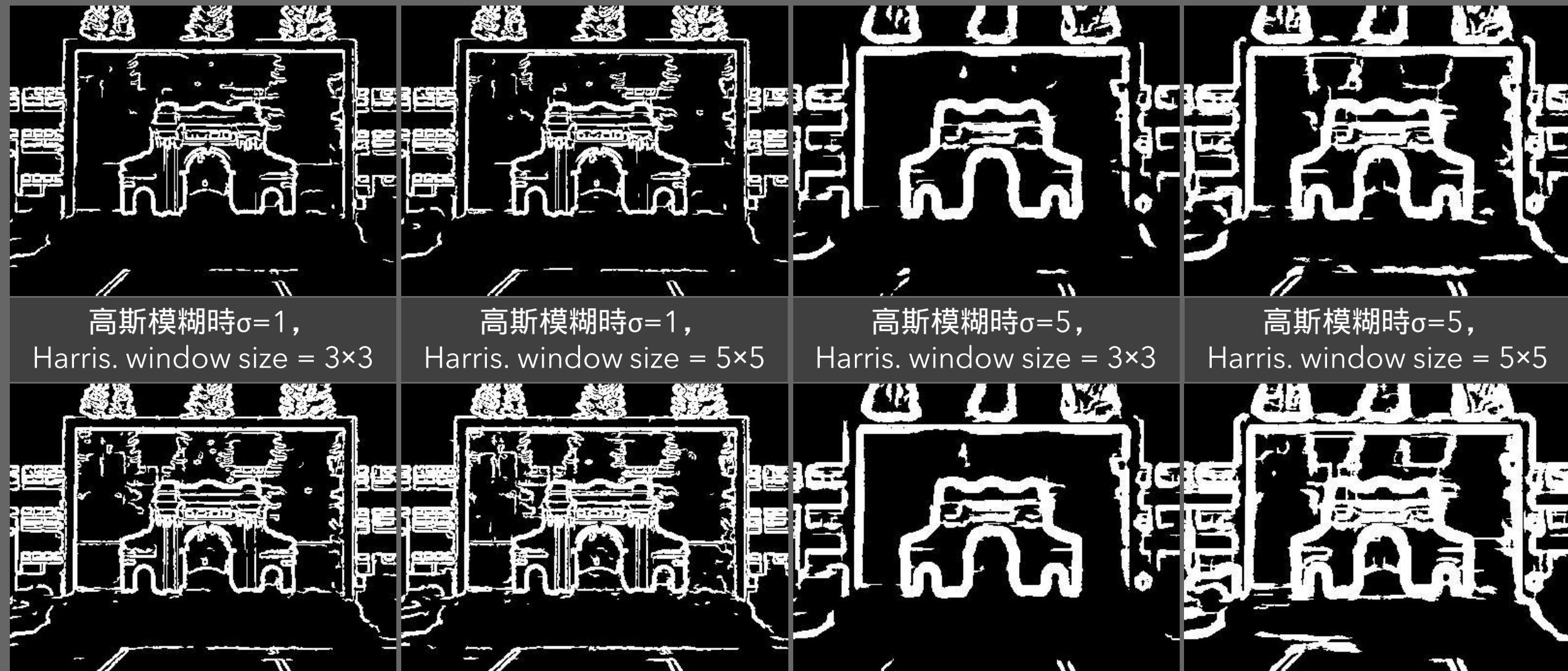


## 2 c, 雪寶：兩種Corner偵測方式，套用Non-maximum Suppression所得的邊緣



上圖以 2a. 的方法偵測的角落中的區域極大值為「種子」，下圖則是2b. 的方法

## 2 c, 清華園：兩種Corner偵測方式，套用Non-maximum Suppression所得的邊緣

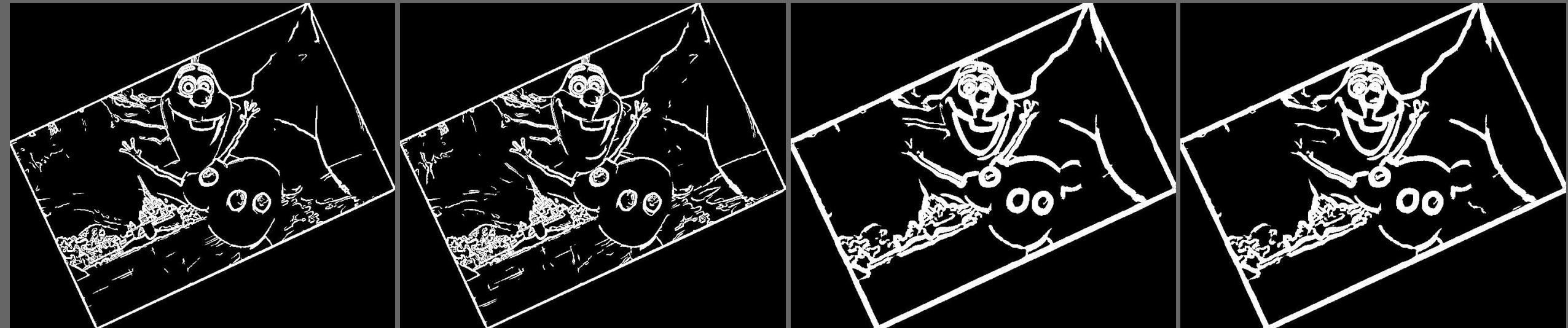


上圖以 2a. 的方法偵測的角落中的區域極大值為「種子」，下圖則是2b. 的方法

## 關於 2c

1. 使用2a. 偵測角落的結果的區域極大值做為種子，和  
使用2b. 偵測角落的結果的區域極大值做為種子，兩種方式以  
non-maximal suppression做邊緣偵測，所得的結果相去不遠，  
因為只要種子有落在邊緣上，通常就能完整地找出邊緣。
2. 上方第1.點提到的兩種方法，因為判斷是否為角落的方式不同，因此  
差別在於以某些「似乎是角落」的位置延伸出去的邊緣會受到影響。  
R (即corner response function)中的k也是影響來源之一。

## 2 d, 雪寶：旋轉和放大後，再次套用Non-maximum Suppression所得的邊緣

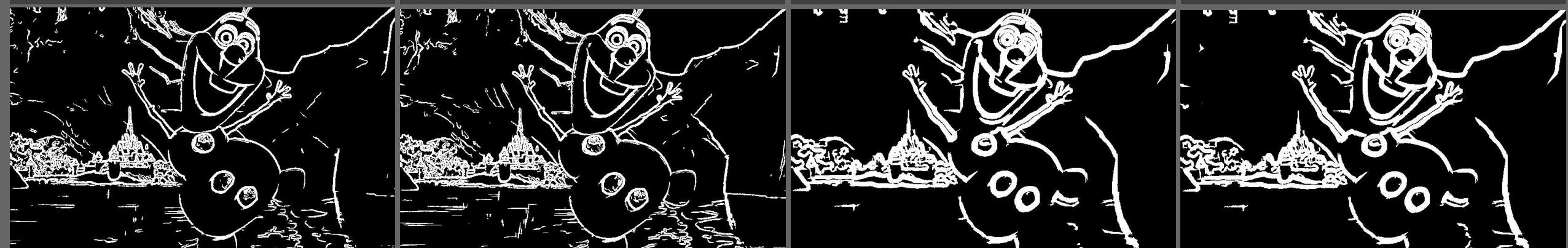


高斯模糊時  $\sigma=1$ ，  
Harris. window size =  $3 \times 3$

高斯模糊時  $\sigma=1$ ，  
Harris. window size =  $5 \times 5$

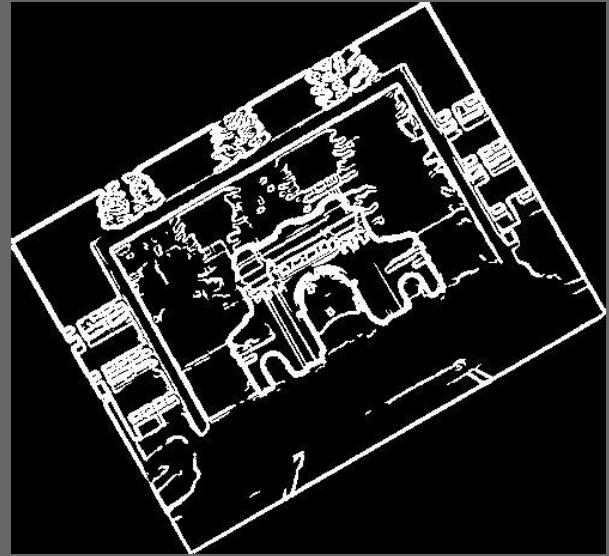
高斯模糊時  $\sigma=5$ ，  
Harris. window size =  $3 \times 3$

高斯模糊時  $\sigma=5$ ，  
Harris. window size =  $5 \times 5$

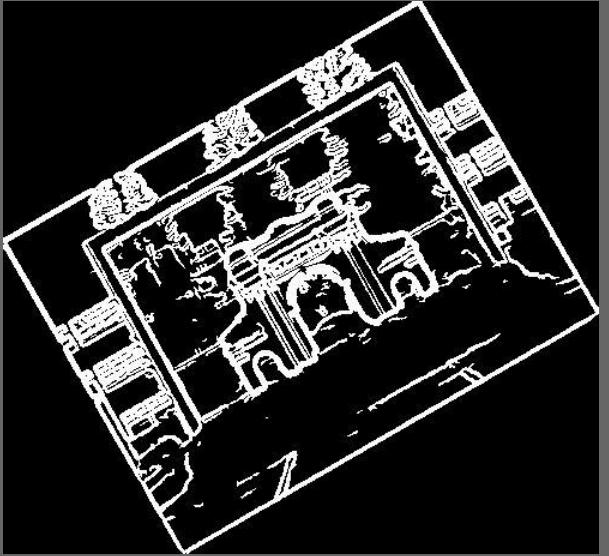


上圖是套用於旋轉的圖片，下圖是套用於放大的圖片

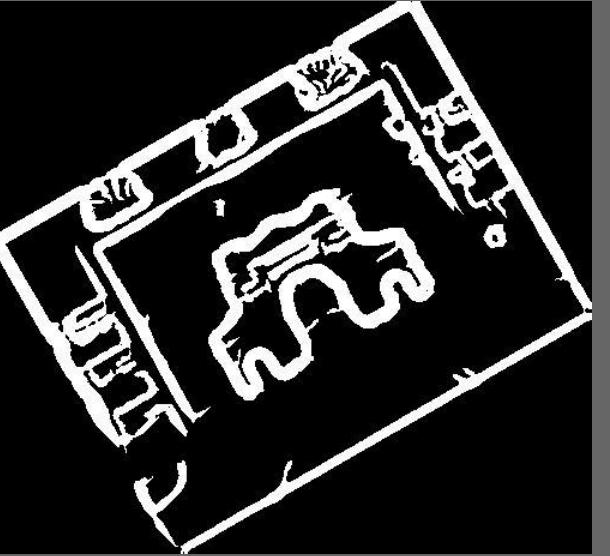
## 2 d, 清華園：旋轉和放大後，再次套用Non-maximum Suppression所得的邊緣



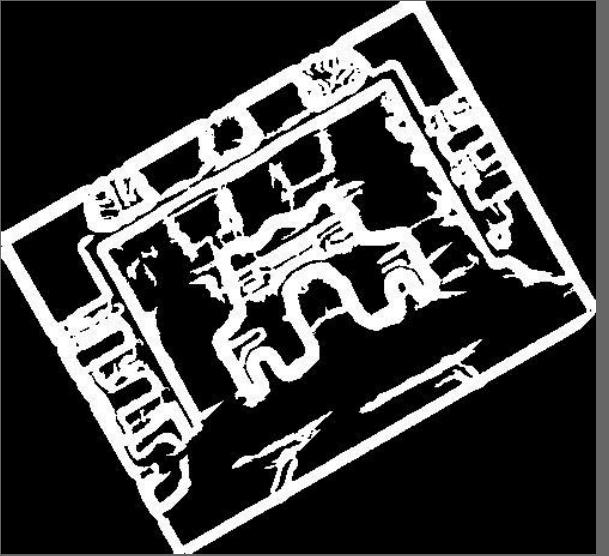
高斯模糊時 $\sigma=1$ ,  
Harris. window size =  $3 \times 3$



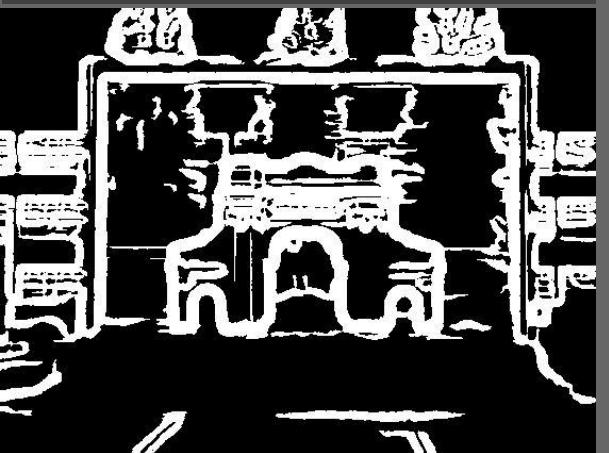
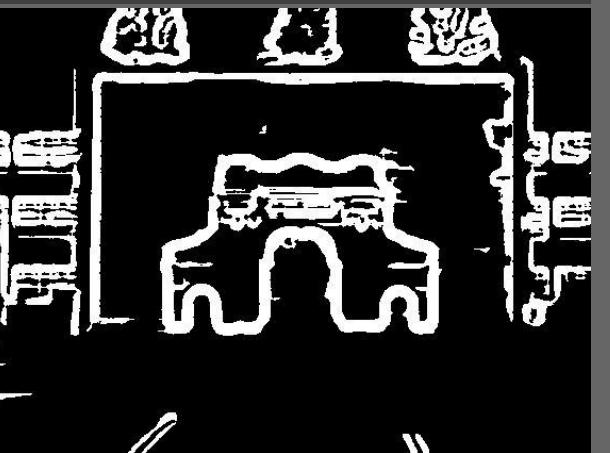
高斯模糊時 $\sigma=1$ ,  
Harris. window size =  $5 \times 5$



高斯模糊時 $\sigma=5$ ,  
Harris. window size =  $3 \times 3$

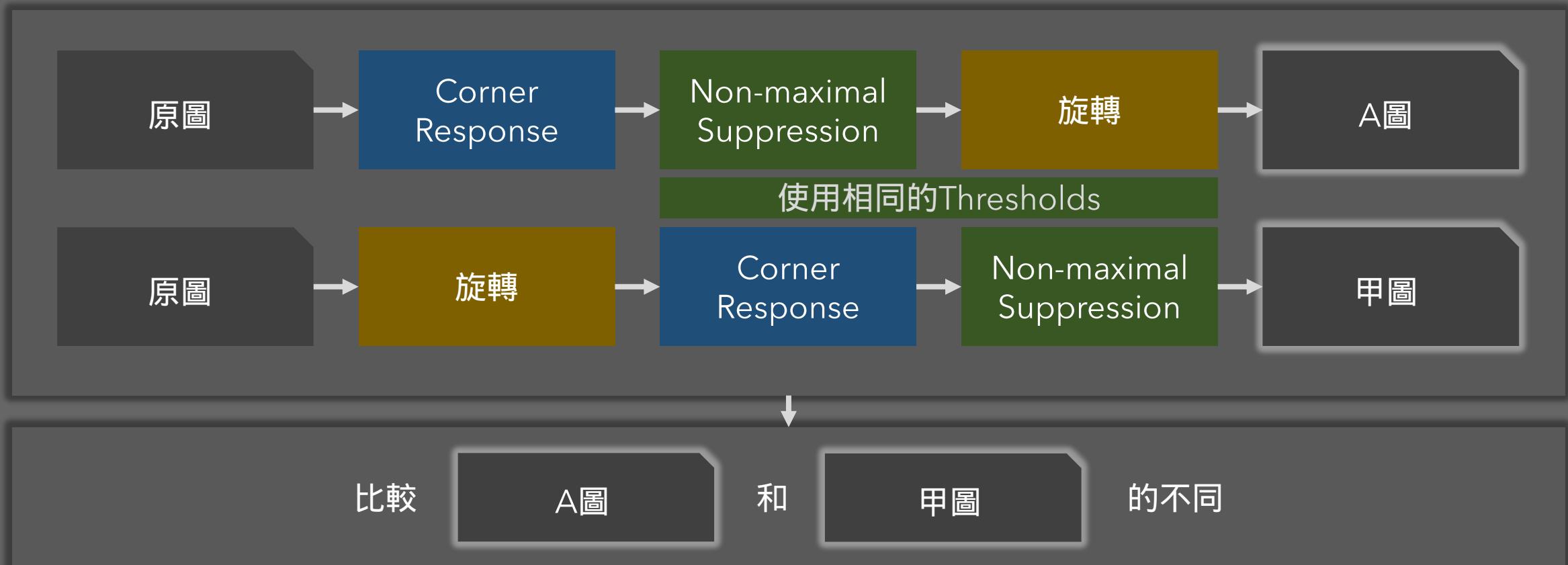


高斯模糊時 $\sigma=5$ ,  
Harris. window size =  $5 \times 5$



上圖以 2a. 的方法偵測的角落中的區域極大值為「種子」，下圖則是 2b. 的方法

## 2e, 實驗流程



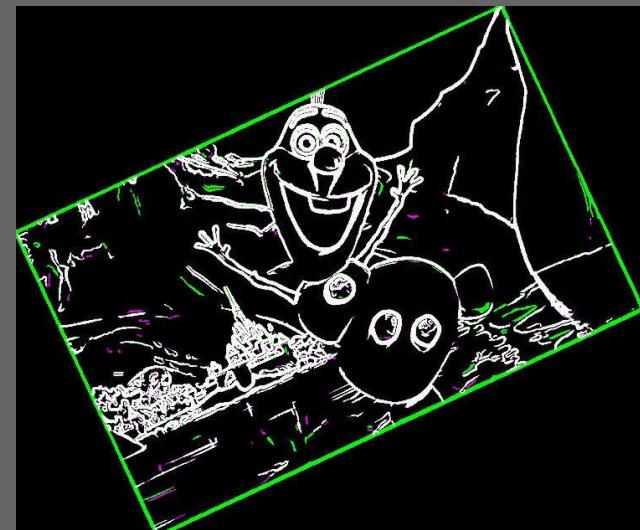
產生A圖和甲圖所用的 Non-maximal Suppression , 其thresholds保持相同

比較縮放的圖片時也使用相同的流程

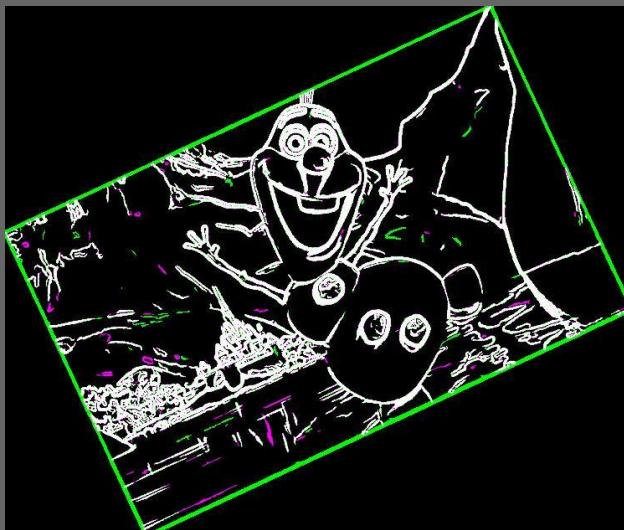
## 2 e, 雪寶：比較邊緣偵測的演算法(Non-maximum suppression)對於旋轉的容忍度

這些圖片表示「A圖」和「甲圖」間的異同處

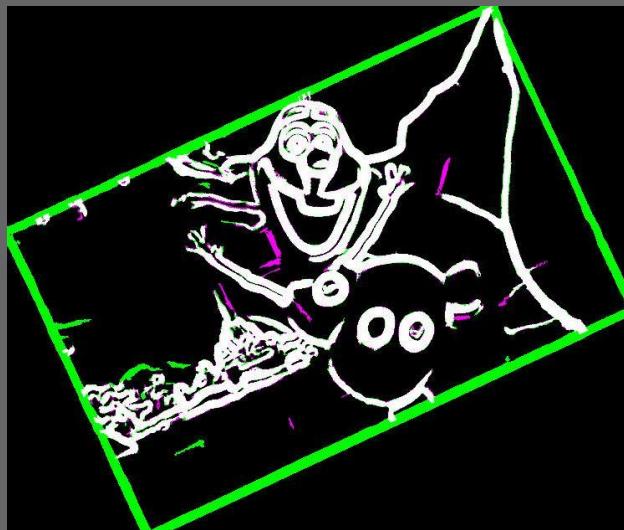
請忽略綠色的斜框框



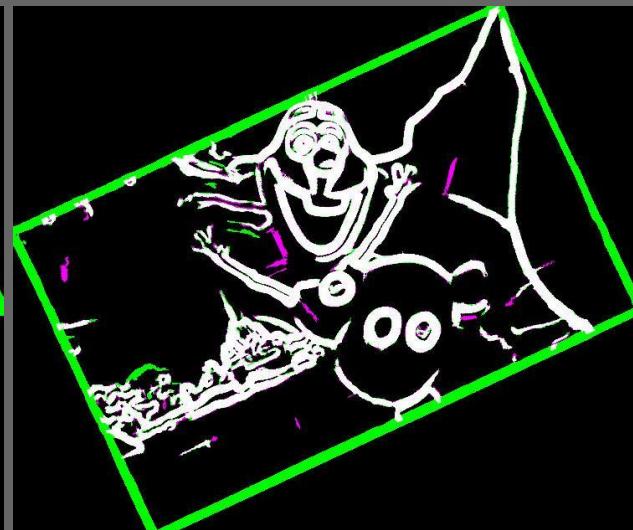
高斯模糊時 $\sigma=1$ ,  
Harris. window size =  $3 \times 3$



高斯模糊時 $\sigma=1$ ,  
Harris. window size =  $5 \times 5$



高斯模糊時 $\sigma=5$ ,  
Harris. window size =  $3 \times 3$

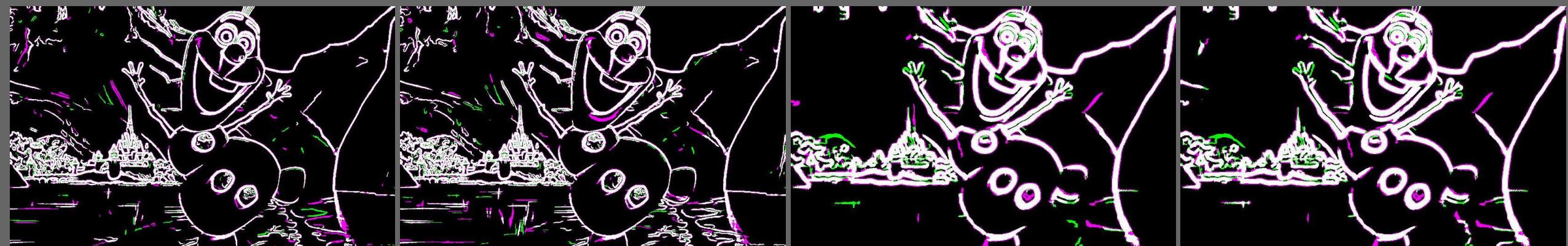


高斯模糊時 $\sigma=5$ ,  
Harris. window size =  $5 \times 5$

綠色的部分是甲圖有，而A圖沒有的邊緣  
紫色的部分是甲圖沒有，而A圖有的邊緣  
白色的部分是兩者都有的邊緣

## 2 e, 雪寶：比較邊緣偵測的演算法(Non-maximum suppression)對於縮放的容忍度

這些圖片表示「A圖」和「甲圖」間的異同處



高斯模糊時 $\sigma=1$ ,  
Harris. window size =  $3 \times 3$

高斯模糊時 $\sigma=1$ ,  
Harris. window size =  $5 \times 5$

高斯模糊時 $\sigma=5$ ,  
Harris. window size =  $3 \times 3$

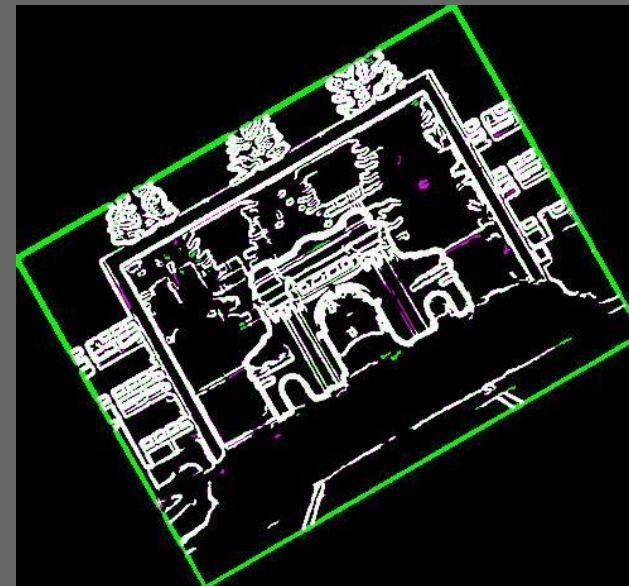
高斯模糊時 $\sigma=5$ ,  
Harris. window size =  $5 \times 5$

綠色的部分是甲圖有，而A圖沒有的邊緣  
紫色的部分是甲圖沒有，而A圖有的邊緣  
白色的部分是兩者都有的邊緣

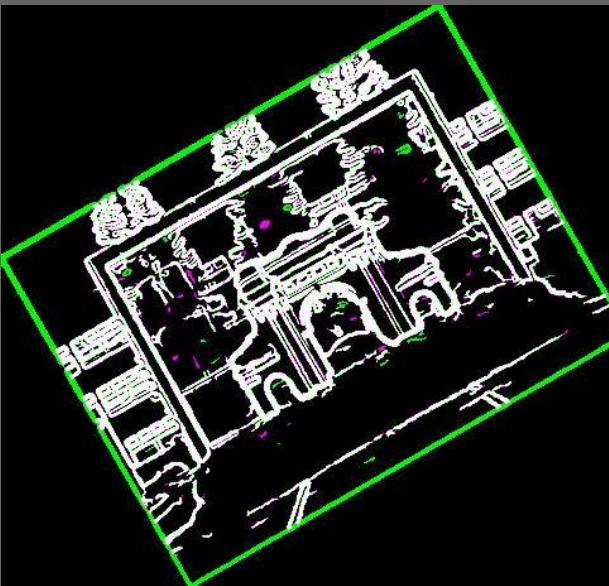
## 2 e, 清華園：比較邊緣偵測的演算法(Non-maximum suppression)對於旋轉的容忍度

這些圖片表示「A圖」和「甲圖」間的異同處

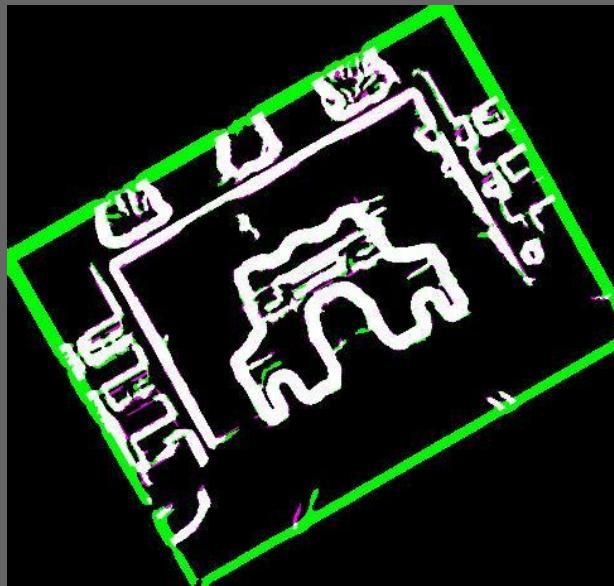
請忽略綠色的斜框框



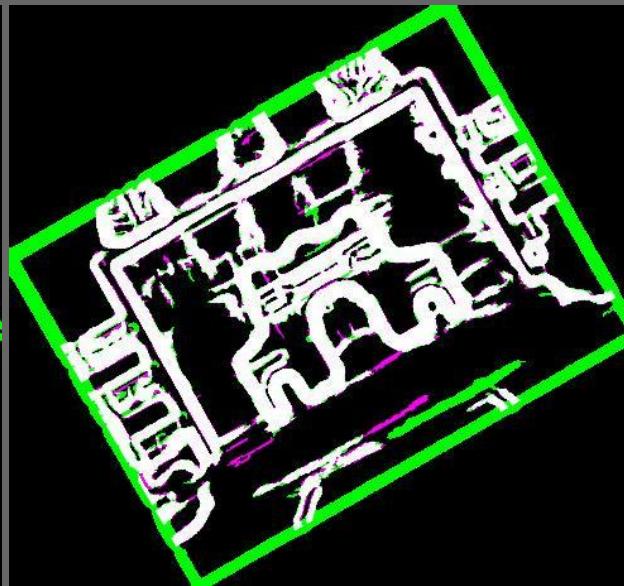
高斯模糊時 $\sigma=1$ ,  
Harris. window size =  $3 \times 3$



高斯模糊時 $\sigma=1$ ,  
Harris. window size =  $5 \times 5$



高斯模糊時 $\sigma=5$ ,  
Harris. window size =  $3 \times 3$



高斯模糊時 $\sigma=5$ ,  
Harris. window size =  $5 \times 5$

綠色的部分是甲圖有，而A圖沒有的邊緣  
紫色的部分是甲圖沒有，而A圖有的邊緣  
白色的部分是兩者都有的邊緣

## 2 e, 雪寶：比較邊緣偵測的演算法(Non-maximum suppression)對於縮放的容忍度

這些圖片表示「A圖」和「甲圖」間的異同處



高斯模糊時 $\sigma=1$ ，  
Harris. window size =  $3 \times 3$

高斯模糊時 $\sigma=1$ ，  
Harris. window size =  $5 \times 5$

高斯模糊時 $\sigma=5$ ，  
Harris. window size =  $3 \times 3$

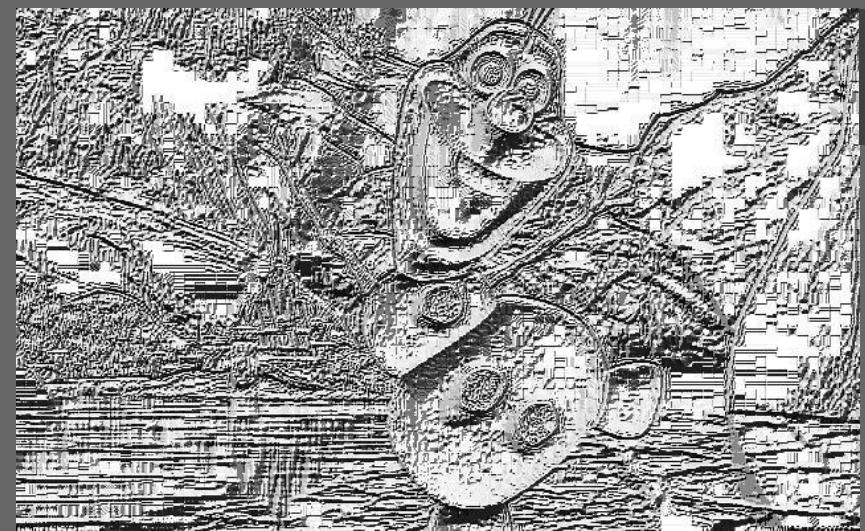
高斯模糊時 $\sigma=5$ ，  
Harris. window size =  $5 \times 5$

綠色的部分是甲圖有，而A圖沒有的邊緣  
紫色的部分是甲圖沒有，而A圖有的邊緣  
白色的部分是兩者都有的邊緣

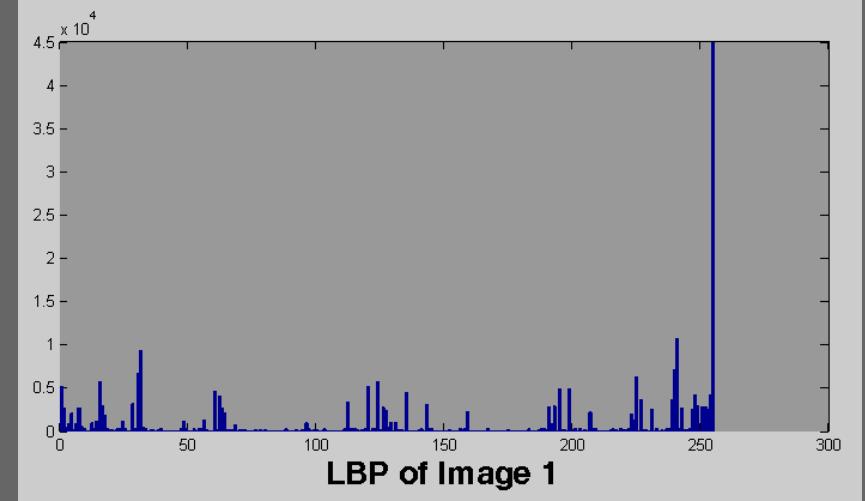
# 關於 2e

1. 整體而言，角落偵測和邊緣偵測的演算法具有一定的consistency，  
您可以看見在結果中，大部分的部分都呈現「白色」。
2. 對於旋轉的比較，紫色部分和綠色部分所占的量大約參半，因此，  
我們可以說基於Bicubic的旋轉可以保留原來影像的邊緣資訊。
3. 對於縮放的比較，圖片放大後資訊量增加了，但是基於Bicubic的放  
大，大體而言保留了稍微多的低頻訊號，將邊緣稍微「模糊化」了，  
因此您可以發現在縮放的比較圖片中，紫色的區域較綠色為多。

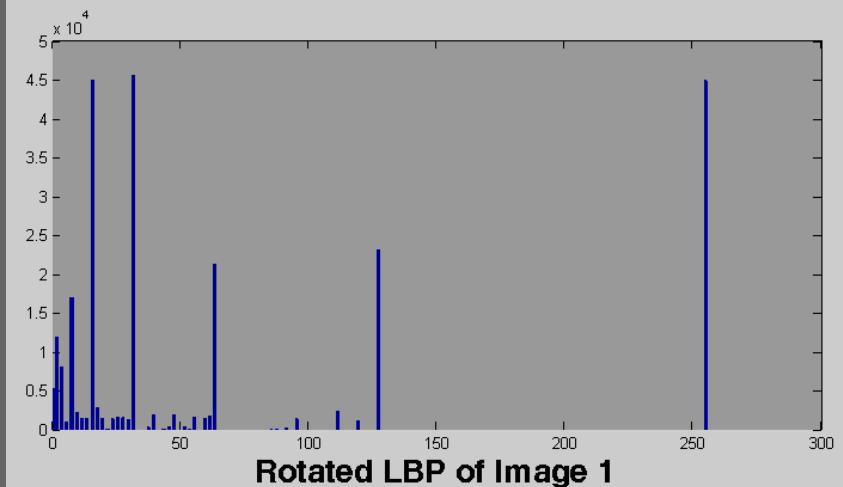
### 3, 雪寶：非位於邊緣的pixel的LBP



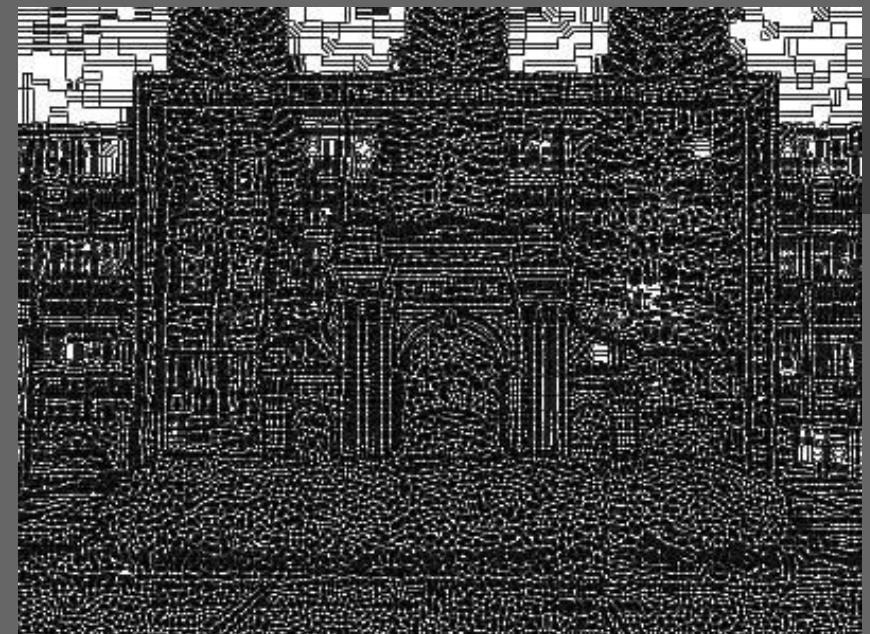
LBP Value at each pixel  
(未旋轉)



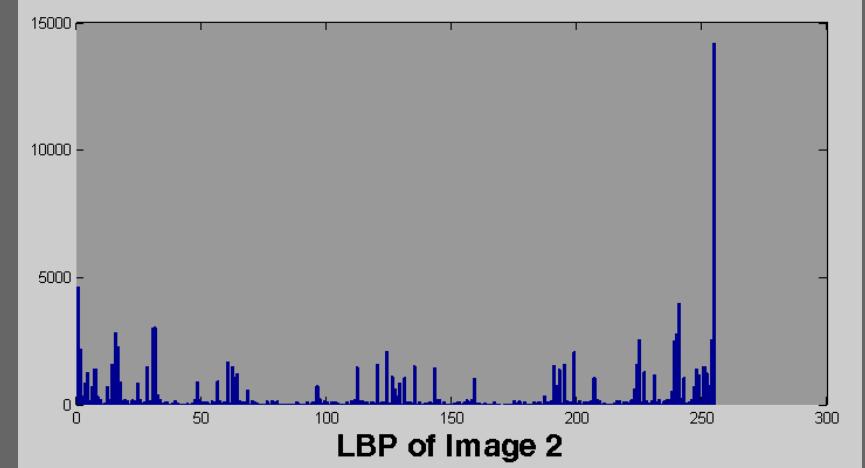
LBP Value at each pixel  
(旋轉7次取其值最小)



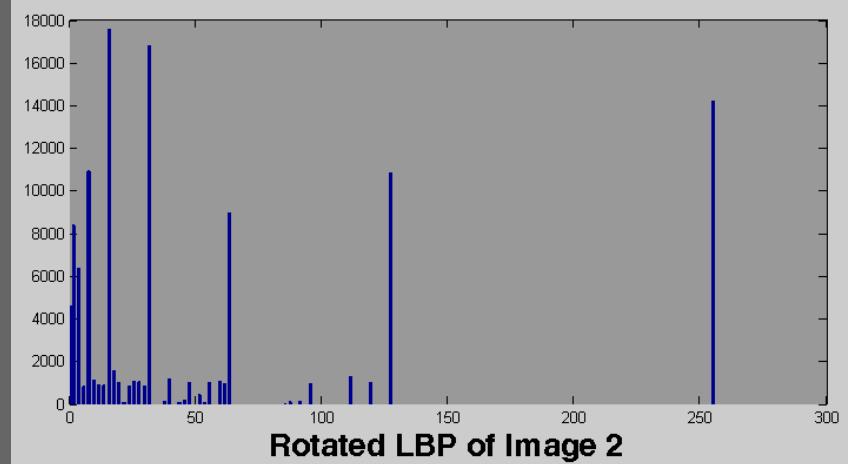
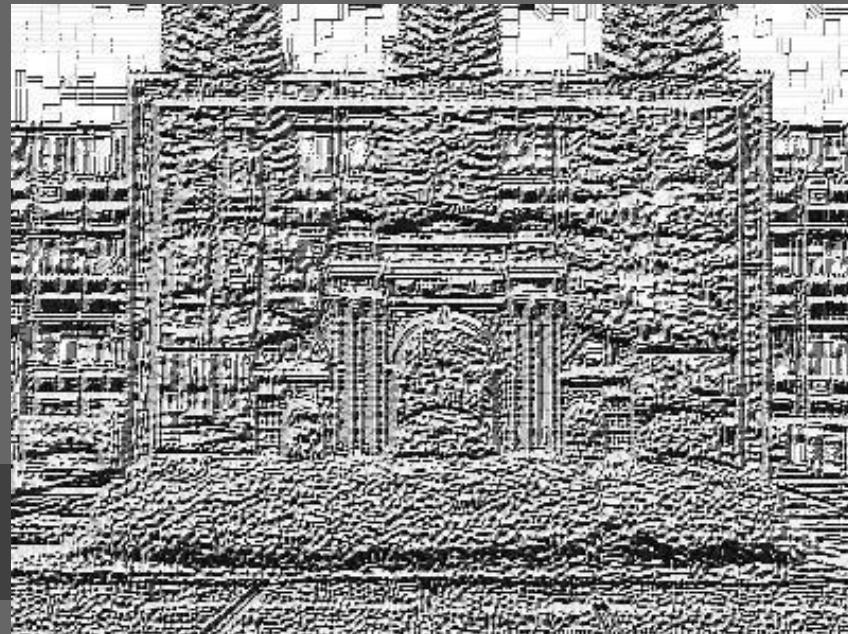
### 3, 清華園：非位於邊緣的pixel的LBP



LBP Value at each pixel  
(未旋轉)



LBP Value at each pixel  
(旋轉7次取其值最小)



# 關於 3

- 您可以發現，在LBP Histogram中，255占約18%，這可能是因為我們將與中心pixel亮度相同的情況給予其bit 1的值，而影像裡可以找到許多區塊，在其中的pixel亮度值完全相同。
- 甚至，在兩張LBP影像中，出現大塊的白色區域。
- 或許這是因為JPEG壓縮導致高頻訊號被削弱，而低頻訊號的強度又不足，所產生的結果。
- 您注意到了嗎？上一頁投影片的兩張影像位置反了。