# Lab11

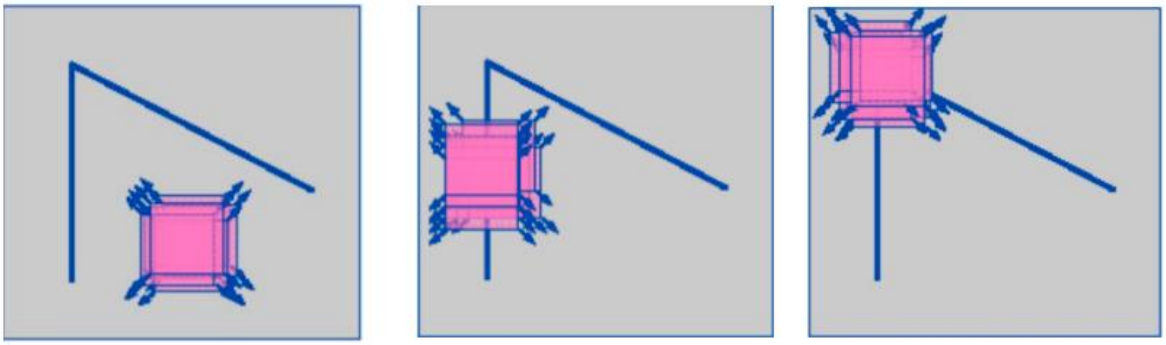
109061217 林峻霆

- ## Lab Objective

  實做 Harris corner detection algorithm 並探討其背後的原理以及解決一些實作中遇上的問題。

- ## Design Implementation

  我們首先解釋 corner detection 的核心想法：我們可以透過觀察一個 window 在 x,y 方向上移動的變化量來判斷這個 window 所處的位置是 corner, edge 或是 flat region。如下圖所示，往兩軸移動都沒有很大的變化→flat region；往某一軸移動會有很大變化→Edge；往兩軸移動都有很大的變化→corner。



Flat region → no change in all directions

Edge → no change along the edge direction.

Corner → significant change in all directions

在數學上，我們可以用微分(或是差分，在數位訊號上)來計算變化量。因此將上述的想法用數學是可以寫成：

$$E(u,v) = \sum_{x,y} w(x,y)[I(x+u,y+v) - I(x,y)]^2$$

這裡的 w(x,y)就是我們取的 window，I 則是我們使用的 image，E(u, v)則是這個 window 經過 shift 後的變化量，後方之所以取平方是因為我們只在乎變化量而不在乎變化的方向。接著，我們可以使用 2D taylor expansion 來簡化我們的分析，2D taylor expansion 如下：

$$\begin{aligned} & f(x_0+\Delta x, y_0+\Delta y) \\ & = f_{(x_0,y_0)} \\ & + f_{x(x_0,y_0)}\Delta x + f_{y(x_0,y_0)}\Delta y \\ & + \frac{1}{2}\left[f_{xx(x_0,y_0)}\Delta x^2 + 2f_{xy(x_0,y_0)}\Delta x\Delta y + f_{yy(x_0,y_0)}\Delta y^2\right] \\ & + O\left(\Delta x^3 + \Delta y^3\right) \end{aligned}$$

若我們只考慮一次微分的 term，我們可以將原先的式子簡化成：

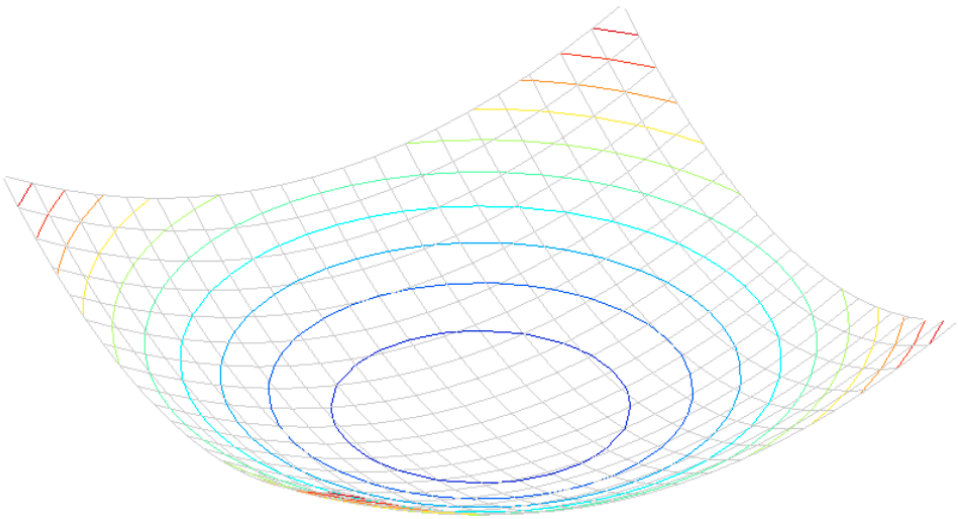
$$I(x+u,y+v) = I(x,y) + uI_x(x,y) + vI_y(x,y)$$

$$\rightarrow E(u,v) = \sum_{x,y} w(x,y)[u^2I_x^2 + 2uvI_xI_y + v^2I_y^2]$$

我們將上面的式子寫成矩陣的形式：

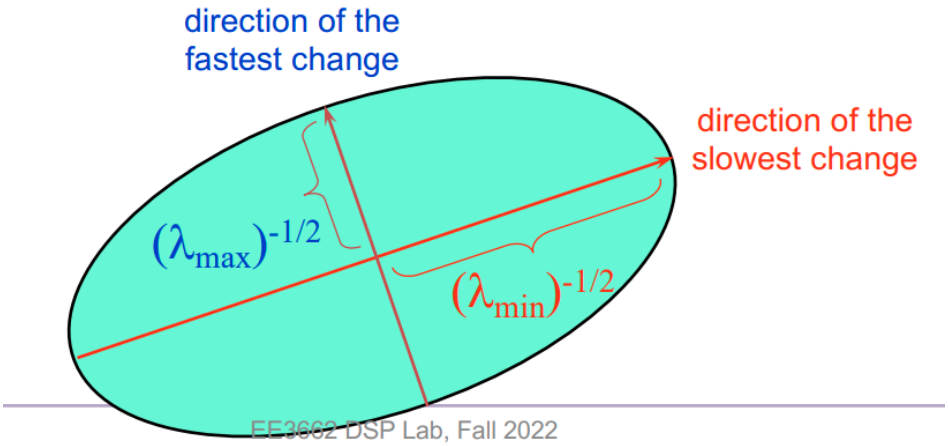
$$E(u,v) = \sum_{x,y}(u,v)\cdot w(x,y)\cdot \begin{pmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{pmatrix}\cdot \begin{pmatrix} u \\ v \end{pmatrix}$$

我們就將$A = \begin{pmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{pmatrix}$定義成 Harris Matrix。那我們要怎麼從這些計算的結果來判斷 corner、edge、flat region 呢？若我們將 E(u, v)畫在一個三維的空間，他會是一個凸函數(如下圖)，我們計算出來的值都對應他的某一個截面



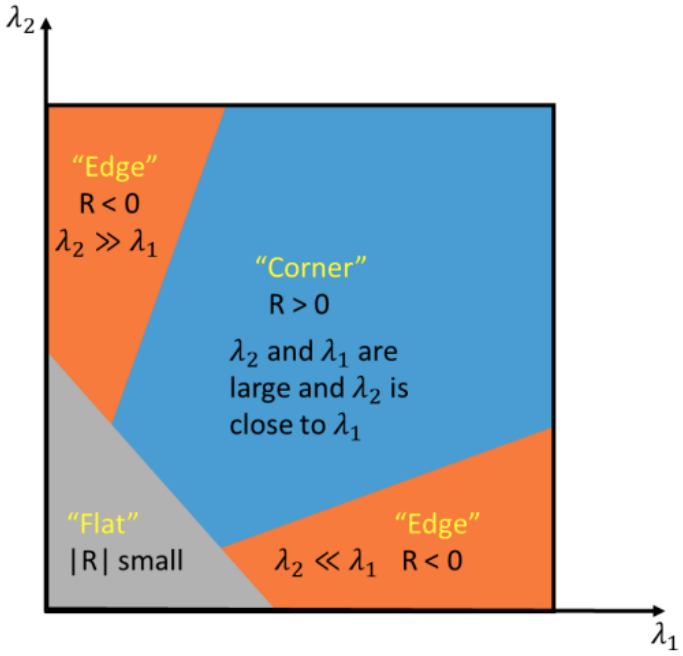
他的每個截面對應到的都是一個橢圓形，這個橢圓形的長軸和短軸由 Harris matrix 的 eigenvalue 來決定，而他跟水平面的夾角(旋轉角度)，則由一個旋轉矩陣 Q 來決定。這個轉矩陣可以透過 Harris Matrix 的矩陣對角化來求得(如下)

$$A = Q^{-1}\begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix} Q \quad where\ \lambda_1, \lambda_2\ are\ eigen\ values$$

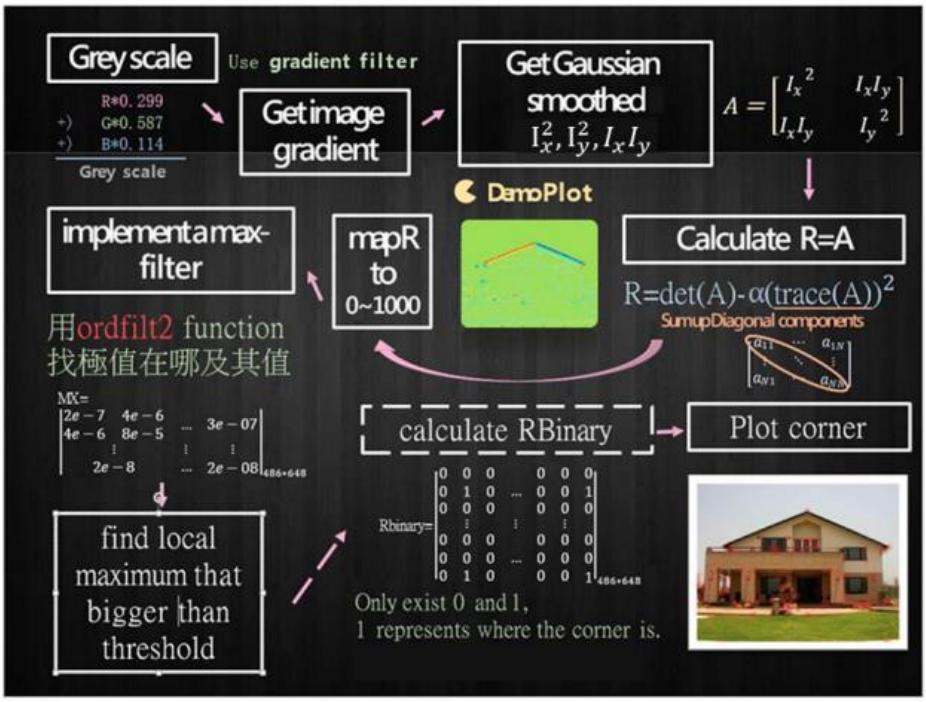



我們可以依靠$\lambda_1, \lambda_2$的大小關係來判斷是 corner, edge 還是 flat region。由於 eigen value 的計算較為複雜，我們也可以定義 corner response R = $\det(A) - \alpha \cdot trace(A)^2$，$\alpha$是一個由經驗法則找出的數值，通常在 0.04-0.06 之間，由 R 的值來判斷是 corner, edge 或 flat region。

兩種判斷方法總結如下：




實務上，我們所實作的 Harris corner detection algorithm 的步驟如下圖：



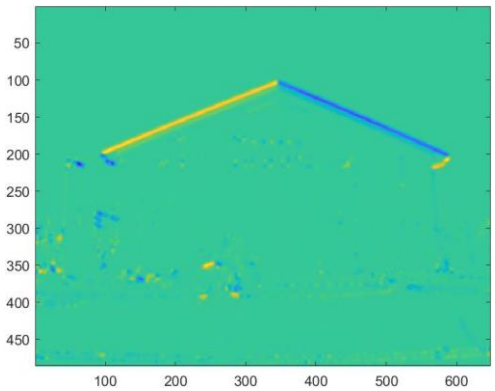

我們首先將圖片轉換成 Gray scale 降低計算複雜度，接著對兩軸計算微分並通過一個 Gaussian filter(一個 LPF，去除雜訊)，然後我們就計算他的 corner response(R)。但是，我們這邊計算出來的 R 值在局部區域可能會有多個滿足上述條件的點，因此我們只取局部的最大值並設定 threshold，來挑選出是當的特徵點當作我們的 corner。實作結果如下：

## 1. House

Original

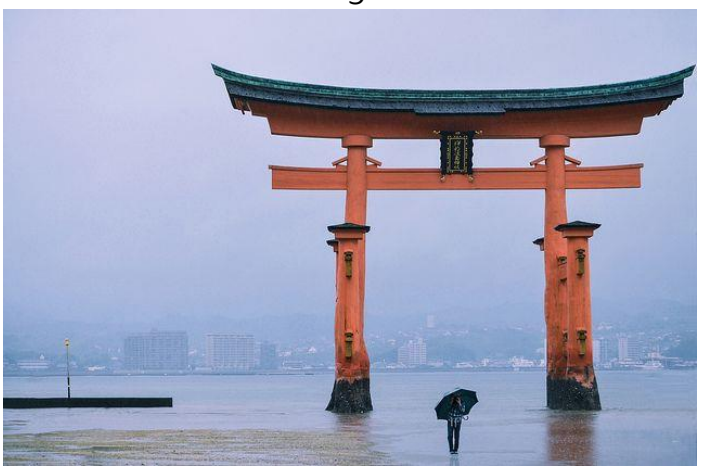
Corner finding

Ixly

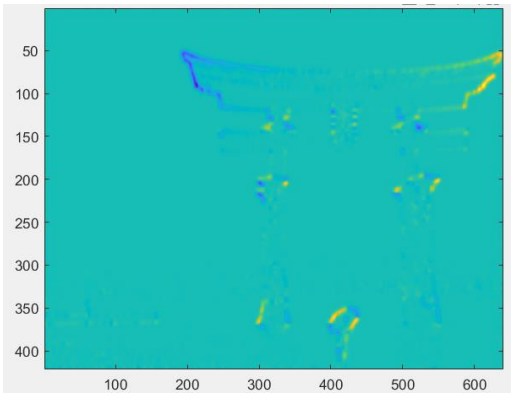



## 2. Torii

Original



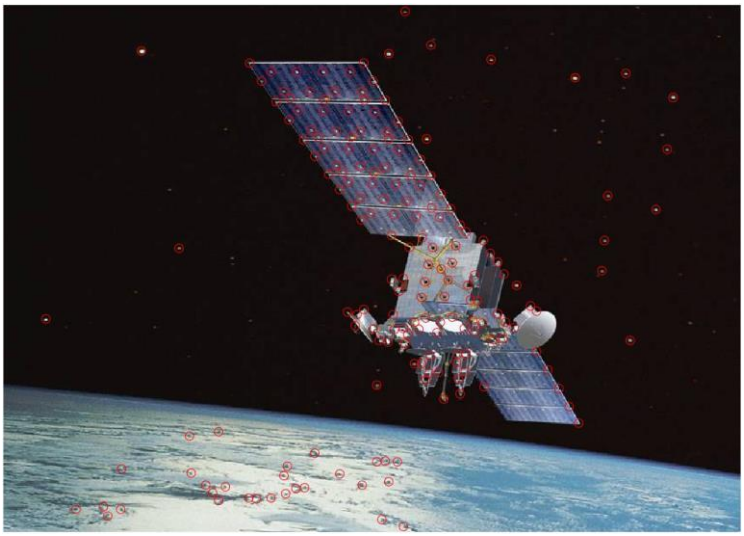
Corner finding Ixly

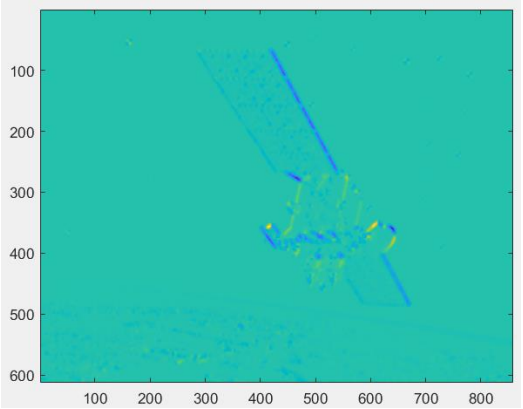



### 3. Satellite

Original

Corner finding Ixly






從結果來看三張圖片中 Harris corner detection 都能有效地找出圖片中物體的 corner，從 Ixly 圖也能看出在圖片中的那些部分會有比較強的變化，在這些區域的 corner response 會比較大，

因此有比較高的機率被選為 corner。另外要注意的一點是我們所 detect 到的點不一定是 corner，從 satellite 的結果來看，只要是跟周圍差異大的點都會被 detect 成 corner，例如背景的星點、地球雲層稀薄處以及衛星上的斑點等。若想要有比較好的結果，可以嘗試調整 threshold、ord2filt 的 size 來讓 corner 只出現在我們認知定義的 corner 上(例如下圖)

Corner detection 可以幫助我們計算物體旋轉的角度，我們用兩個正方形來 demo 這個功能。我們分別對這兩個正方形做 corner detection，然後運用他的四個 corner(邊角)來計算他與水平軸的夾角(如下圖)，進行相減就能得到這個正方形旋轉多少角度。


實作結果如下：

```
Rotation Degree: 2.364868e+01
```

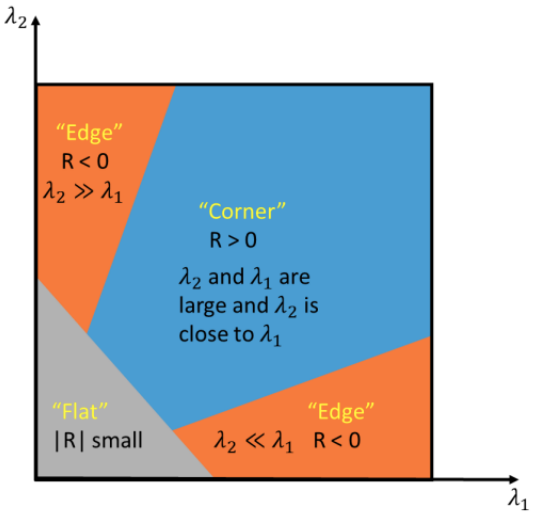
- ## Discussion

### 1. Implementation of Edge Detection

根據我們前面的推導，只要將 algorithm 經過些微修改，我們也能進行 edge detection。以下是我所做的修改：

1. ord2filt 找範圍中最大的值作為代表 → 最大的作為代表
2. R > threshold → R < threshold
3. 不將 R 值 map 到 0 ~ 1000，採用原先的 R 值

要注意的是，這邊的 threshold 必須要是負數。原因可以參考下圖，R 值小於 threshold 的位置就是 edge，為了避免連同 flat region 也被 detect 出來，我們不採用 0 作為 threshold，而是採用一個夠小的負數(ex. -0.1, -0.01, -0.005 之類的就足夠去除 flat region，可根據圖片做調整)

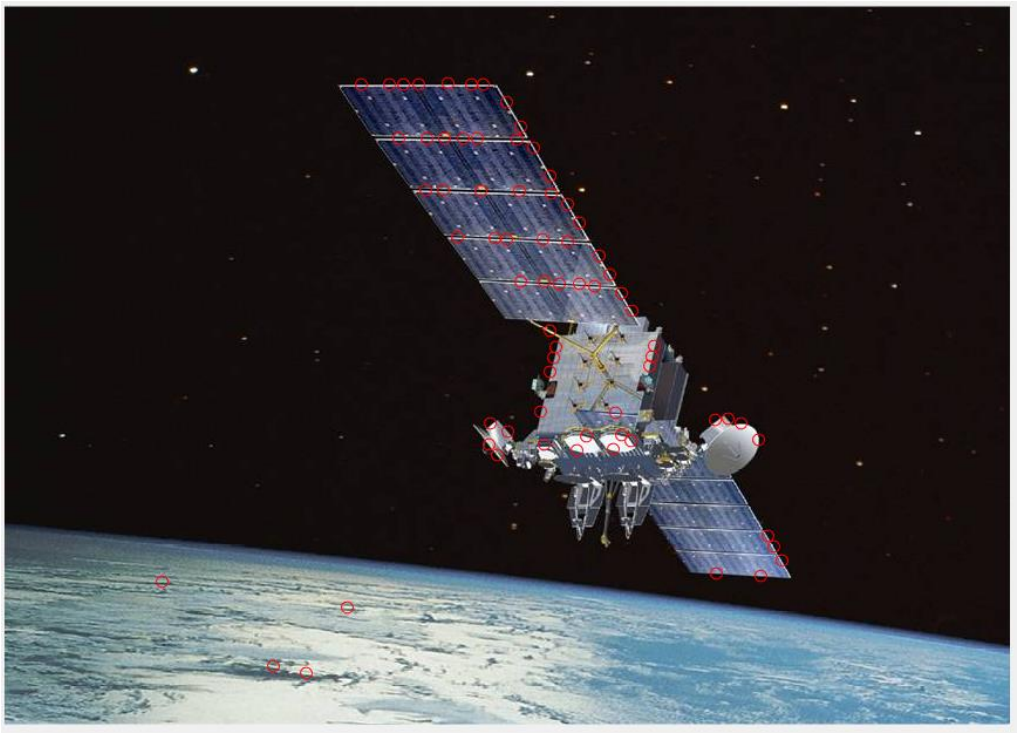



實作結果如下：

1. House

2. Torri

3. Satellite



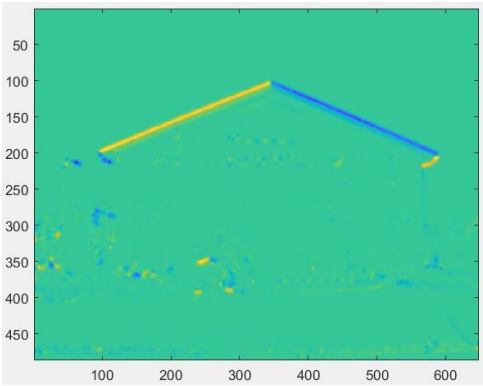
## 2. Windowing Issue

這部分我們嘗試不同種的 smoothing window 並比較其差異。我所嘗試的 smoothing window 有 gaussian window, rectangular window。Rectangular window 如下：

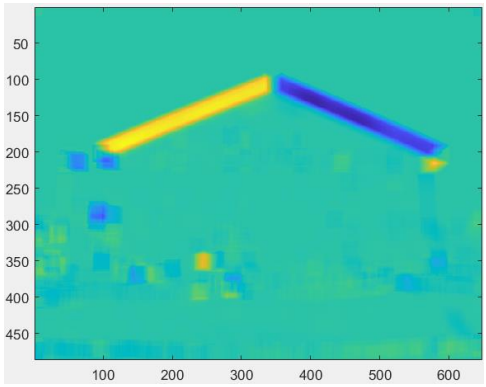
| | | | |
|---|---|---|---|
| 1 | 1 | ... | 1 |
| 1 | 1 | ... | 1 |
| ... | ... | ... | ... |
| 1 | 1 | ... | 1 |

實作的結果如下：

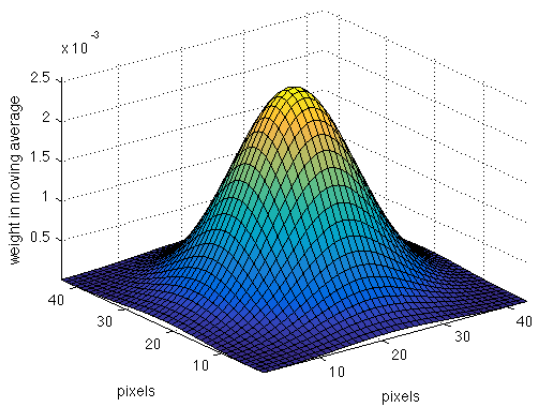
1. Gaussian window

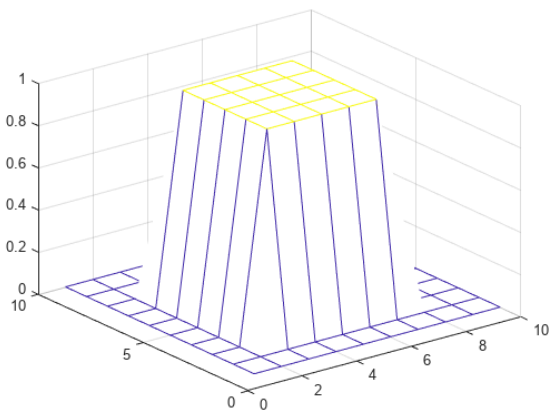



2. Rectangular window



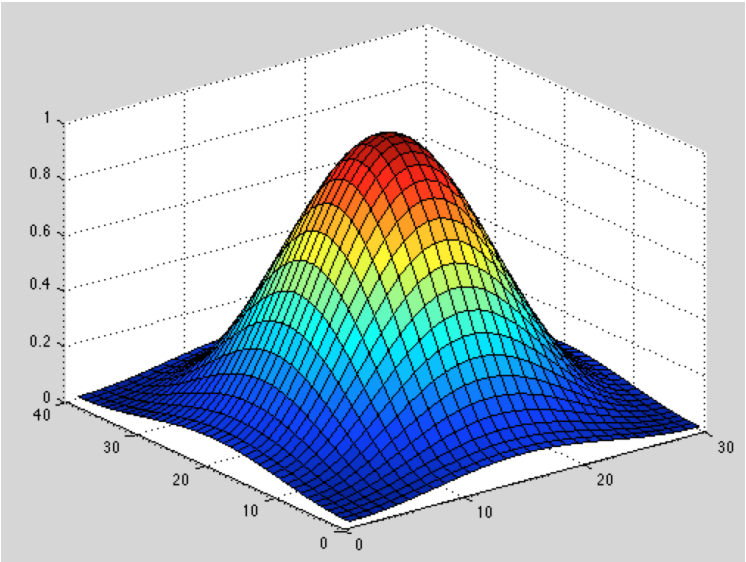

可以發現 rectangular window 所標出來的 corner 位置不太正確，出現了飄移。我認為原因可以從 IxIy 的圖來解釋，可以發現 rectangular window 的 IxIy 中 response 較為強烈的位置的範圍相較於 Gaussian window 的範圍寬，原因在於兩個 filter 的形狀不同(參考下圖)，gaussian filter 越靠近中心佔的權重越大，因此經過 smoothing 後 response 強的點不會飄移，而 rectangular filter 則會造成最高點飄移的現象。



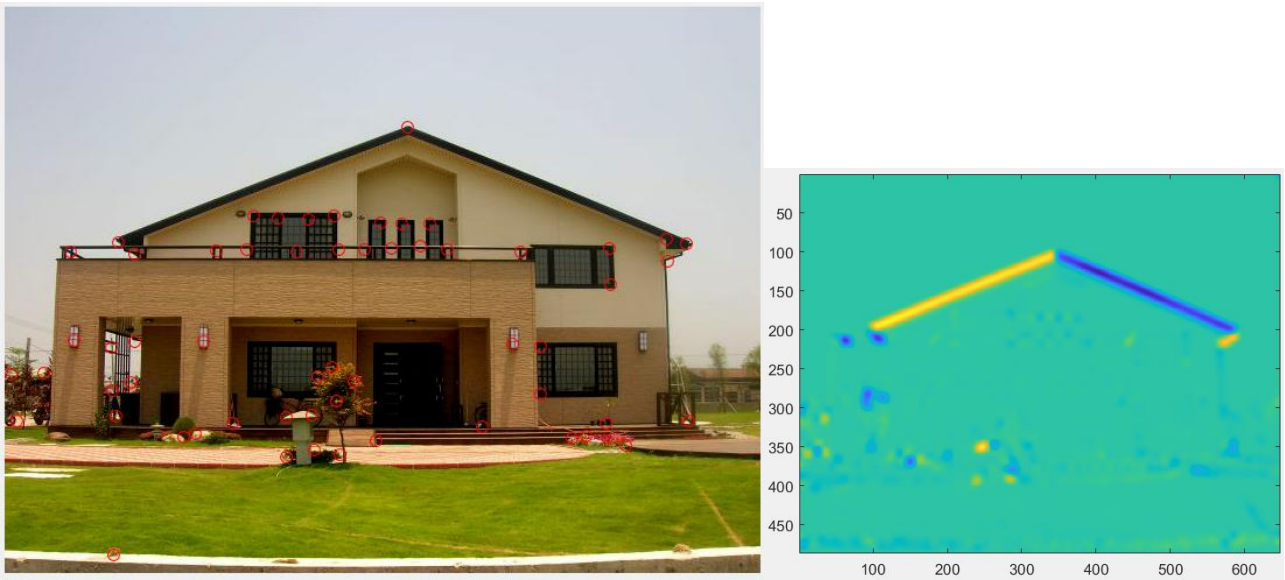



由以上結果可知，我們應該挑選一個類似有鐘形曲線的函數，所以我嘗試另一種 window function：2D hanning window，他長得如下圖所示。選這個 window 的原因在於他跟 gaussian window 相似都有在中間較高，兩側較低的趨勢，這樣能再去出雜訊的同時保持最高點不會飄移。




實作結果如下：

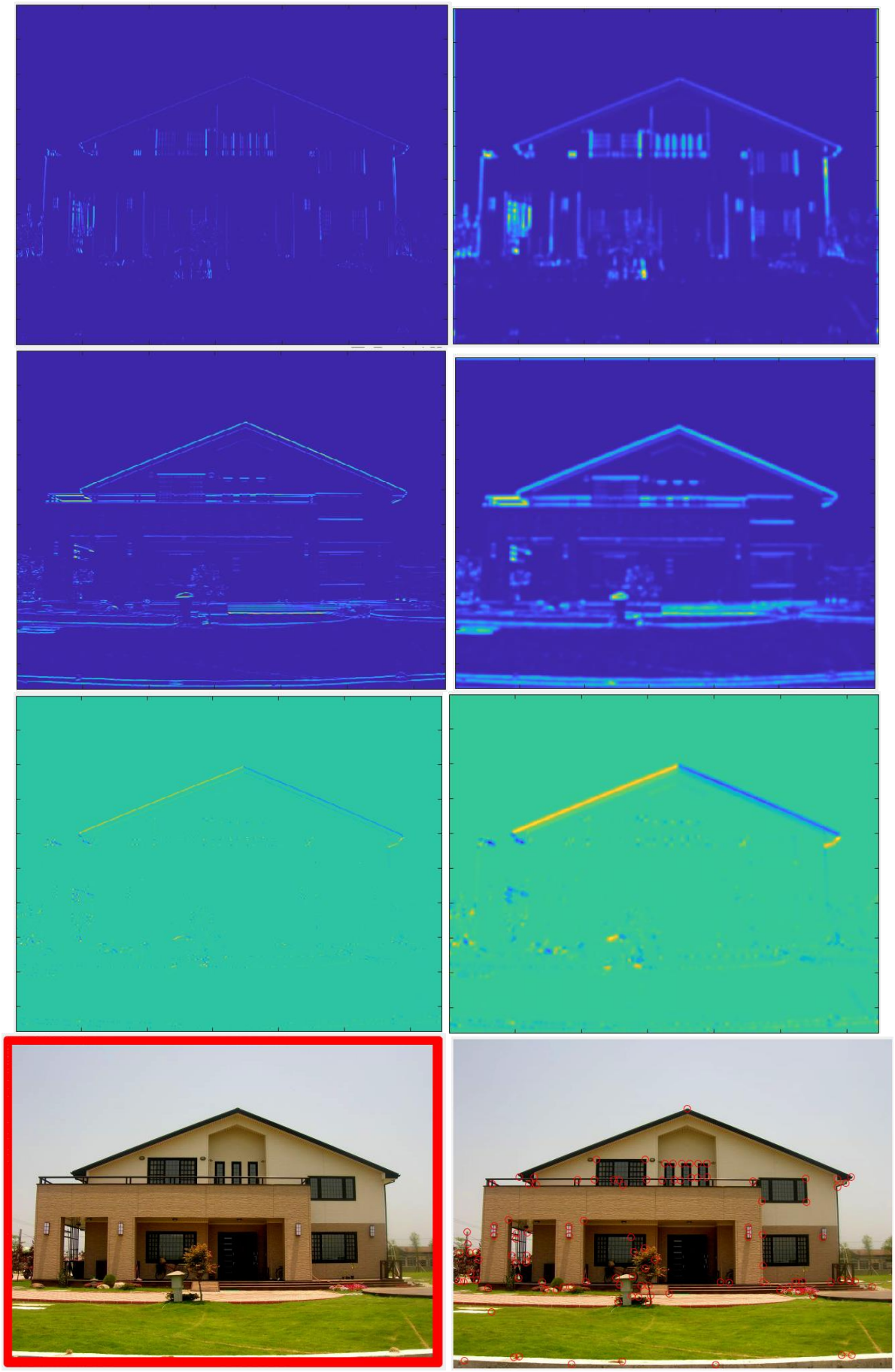



可以看到相較於 rectangular window，hanning window 的結果較符合我們的預期，detect 出來的 corner 位置較為準確，然而相較於 gaussian window 而言 performance 仍然差了一些，detect 出的 corner 數量較少。

### 3. Smoothing problem

這部分我們來探討 Smooth filter 的功用。Smooth filter 是一種 low pass filter，如前面所述，他的作用在於消除雜訊，但$I_x^2$, $I_y^2$, $I_xI_y$的雜訊來自何處呢？我們先看看如果不用 smoothing filter 的結果(上到下分別為$I_x^2$, $I_y^2$, $I_xI_y$以及 detection result)：

No smooth Smooth



可以發現 detect 到的 corner 全部都在圖片的邊界。原因在於我們前面在做微分時，我們是使用一個 gradient filter 去與圖片做 convolution(使用 matlab 的 imfilter)，matlab 的

imfilter 為了讓 convoltuion 前後的圖片大小不變會在圖片周圍做 zero padding，導致圖片的邊界出現高頻雜訊(一邊為 0 一邊有數值)。這會導致在計算微分時，圖片邊界會有很大的變化量，但這其實不是我們要找的 corner。這時，我們使用一個 smooth filter 就能消除這些在邊界因為 zero padding 出現的高頻成分，進而讓我們的演算法能正確找到我們要的 corner。另外，從$I_x^2$, $I_y^2$, $I_xI_y$三圖的比較也能看出，加了 smooth filter 後，更多原先看不到的低頻成分能夠顯現出來。

## 4. Different gradient filter

以下我們嘗試用另外兩種 gradient filter(Prewitt filter and Scharr filter)去做 corner detection，他們分別長得如下：

Prewitt

| dx | | |
|---|---|---|
| 1 | 0 | -1 |
| 1 | 0 | -1 |
| 1 | 0 | -1 |

| dy | | |
|---|---|---|
| 1 | 1 | 1 |
| 0 | 0 | 0 |
| -1 | -1 | -1 |

Scharr

| dx | | |
|---|---|---|
| 3 | 0 | -3 |
| 10 | 0 | -10 |
| 3 | 0 | -3 |

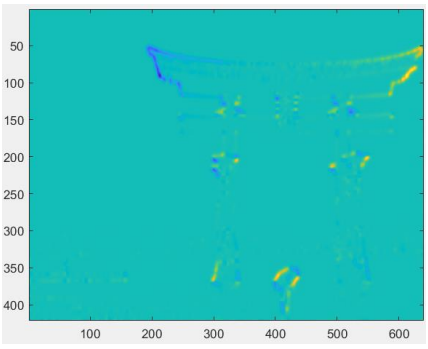
| dy | | |
|---|---|---|
| 3 | 10 | 3 |
| 0 | 0 | 0 |
| -3 | -10 | -3 |

而我們原先用得 gradient filter 則是 sobel filter：

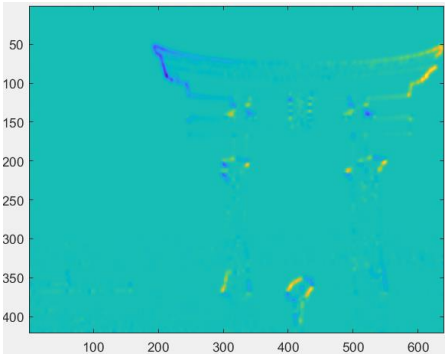
$$G_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \quad G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

實作結果如下：

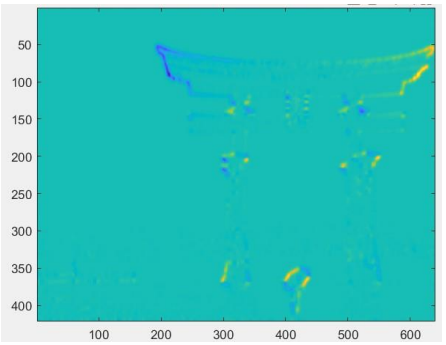
### 1. Prewitt



2. **Scharr**




3. **Sobel**




從 corner detection 的角度我認為三者的 performance 相差無幾，基本上都能有效地找出圖片中的 corner 且數量相近，因此我去比較三種 filter 在 edge detection 的效果：

1. **Prewitt**

2. Scharr

3. Sobel

從 edge detection 的角度就能很清楚的看出三者的差異，從 detect 出來的 edge 相比，scharr > sobel > prewitt。我認為這之間差異的原因可以從三者的 kernel matrix 來解釋，將三者的 kernel matrix 經過 normalization 後，可以發現矩陣中心上下(左右)兩側的數值大小關係是 scharr(10 / 3) > sobel(2) > prewitt(1)。convolution 在做的事情竟是對多個資料點去進行 weighting 並作相加，weighting 越大代表在新產生的資料點中貢獻越多。而上下(左右)兩側的數值大就代表當上下(左右)兩個 pixel 的差距越大時產生的 response 就貢獻越多，而會產生「上下(左右)兩個 pixel 的差距大」的位置就是 edge，這也就是為什麼在相同 threshold 下，三個 filter 的 detection result 有數量差異的原因。

- **Conclusion**

這個 Lab 中我運用助教給的圖以及自己找的圖片實做 corner detection algorithm。運用數學分析的方式去偵測我們預期中 corner 以及 edge 處應該要出現的 response。過程中我們也討論不同的 smoothing window 以及 gradient filter 對於 performance 的影響。此外，我們也

使用實作出來的 corner detection algorithm 去實際解決圖片旋轉角度計算的問題。這個技術或許可以應用在上一個 Lab 中為了產生效果好的 Hybrid image 中所遇到的對齊問題。

- References
  - 教授與助教的講義
  - 2D taylor expansion taylor2dSlides.pdf (ubc.ca)
  - Intro to Harris corner detection
    Introduction to Harris Corner Detector | by Deepanshu Tyagi | Data Breach | Medium
  - Gradient filter comparison Comparing Edge Detection Methods (nikatsanka.github.io)
  - Prof. Chung Yung-Yu's handout
    Microsoft PowerPoint - lec06_feature.pptx (ntu.edu.tw)
  - Intro to window function Window function - Wikipedia