

# Digital Visual Effects – Image Stitching

R13631011 陳冠廷 Team25

Department of Biomechatronics Engineering, National Taiwan University, Taipei, Taiwan

## ABSTRACT

作業相關重點說明：

1. **Baseline**：Feature detection, Feature matching, Image matching and Blending. (60%)
2. **Bonus**：More than one feature detection or description (7%) -> 2.2.1 + 2.2.2
3. **Bonus**：Rectangling (3%) -> 2.5
4. **Bonus**：End-to-end alignment(5%) -> 2.4

**Keywords:** Image Stitching, Feature Detection and Matching, SIFT, Harris Corner, RANSAC, Image Matching and Blending, Cylindrical Projection, Rectangling

## 1. INTRODUCTION

影像拼接是一種透過註冊、變形、重採樣和融合將一組影像結合成一張較大影像的技術。常見應用是全景圖的創建。一般而言，影像拼接有兩種主要方法：直接方法和基於特徵的方法，本研究使用基於特徵的方法。

在本次研究中，我實現《Recognising Panoramas》[3] 論文中的部分內容，主要包括以下四個部分：

1. 特徵檢測 Feature Detection
2. 特徵匹配 Feature Matching
3. 影像匹配 Image Matching
4. 融合 Blending

特徵匹配有兩種選擇，SIFT (`image_stitching_sift.py`)、Harris's (`image_stitching_harris.py`)，分為兩個程式撰寫，除了特徵匹配外，拼接流程皆相似。SIFT較難實現(`sift_impl.py`)，有參考《Distinctive Image Features from Scale-Invariant Keypoints》[4] 及《Implementing SIFT in Python: A Complete Guide》[6]，將參考程式執行速度優化為原本的四倍，並使用了 `ptqt5` 做了 SIFT 的步驟視覺化 UI (`sift_visualizeUI.py`)。

對於焦距的估計，使用 **Autostitch** 取得 `pano.txt` 中的焦距。

這兩個程式用於三腳架拍攝的測試數據拼接相當成功，但在我自己拍攝的數據則出現明顯錯誤，僅在使用兩張圖片拼接時才沒有出現問題，原因應為程式設計數據拍攝時需固定轉動角度，而我拍攝時並沒有固定角度旋轉。



**Image Stitching**



**Rectangling**



Figure 1. 測試數據 /grail 拼接步驟視覺化



**Image Stitching**



**Rectangling**



Figure 2. 測試數據 /parrington 拼接步驟視覺化

## 2. METHODOLOGY

### 2.1 Preprocessing

使用 iPhone 15 手機拍攝不同角度之圖片，並經過由 [HEIC 轉 JPG 檔案](#)，[縮放圖片像素及大小](#)

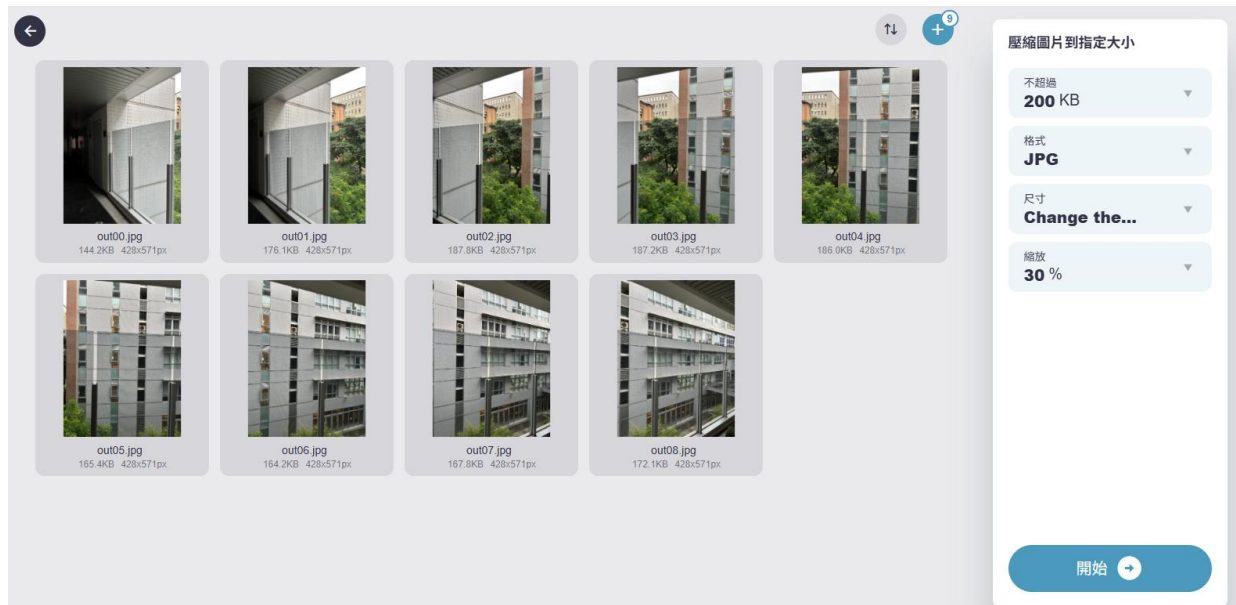


Figure 3. 該程式能批量轉換影像大小到固定像素或縮放，能比直接使用 opencv resize，保留更多細節，且不會破壞影像



Figure 4. 完整的拍攝數據集，從每張大約 3MB 壓縮至 200KB 內，但最終只使用前兩張做拼接，產生最終結果圖



## 2.2 Feature Detection

### 2.2.1 SIFT (Scale-Invariant Feature Transform，尺度不變特徵轉換)

以下說明將完整闡述程式中所用到的 SIFT 核心公式與演算法步驟，包含從金字塔生成、特徵點偵測、極值精細化、方向分配，一直到描述子計算的整體流程。

#### 一、金字塔 (Scale-Space) 與基底影像

##### 1. 生成基底影像 (Generate Base Image)

###### - 上採樣 (Upsampling)

為了增強影像在小尺度處的特徵穩定性，SIFT 會將輸入影像先放大 2 倍。此舉能在後續做多尺度檢測時，維持較高解析度並確保精度。若原圖尺寸為  $(H, W)$ ，上採樣後尺寸變為  $(2H, 2W)$ 。

###### - 初始高斯模糊 (Initial Gaussian Blur)

令上採樣後的影像初始模糊量為  $\sigma_{\text{assumed}}$ （通常取 0.5）。為了得到理想的  $\sigma$ （常見為 1.6），需再做一個「補差量」的模糊，使最終等效模糊量符合需求。

若期望最終高斯模糊標準差為  $\sigma$ ，而已知目前等效模糊為  $2 \times \sigma_{\text{assumed}}$ （因上採樣），補差量即

$$\sigma_{\text{diff}} = \sqrt{\sigma^2 - (2 \times \sigma_{\text{assumed}})^2}$$

接著對上採樣後影像再做一次高斯平滑，最終得到基底影像。



Figure 5. 基底影像

#### 二、計算金字塔層數與生成高斯模糊金字塔

##### 1. 計算金字塔層數 (Compute Number of Octaves)

SIFT 將影像在空間中不斷下採樣 (Downsampling) 形成多個八度 (octave)，直到影像邊長無法再繼續有效縮小為止。

簡化的做法是：

$$\text{num\_octaves} = \lfloor \log_2(\min(H, W)) \rfloor - 1$$

其中  $\min(H, W)$  是基底影像較短的邊長。

## 2. 生成各層高斯核 (Generate Gaussian Kernels)

- 尺度間隔 (scale intervals)

在一個八度 (octave) 內，會額外再分為數個尺度 (scale)。例如設定每個八度有  $\text{num\_intervals}$  個子尺度，SIFT 通常多加 3 張影像以利做差分，故每個 octave 實際上有  $\text{num\_intervals}+3$  張高斯模糊後的影像。

- 幾何級數增長

設  $k = 2^{\frac{1}{\text{num\_intervals}}}$ ，則第  $i$  張影像的等效模糊標準差為  $\sigma_i = \sigma_0 \times k^i$

但實際實作時常儲存「相鄰尺度間需要額外增量的  $\sigma$ 」，其計算來自  $\sigma_{\text{diff},i} = \sqrt{\sigma_i^2 - \sigma_{i-1}^2}$

## 3. 生成高斯金字塔 (Generate Gaussian Images)

- 在同一個八度 (octave) 內做多尺度模糊

首先將基底影像視為  $\sigma_0$  下的起始狀態。依序對同一張影像再做高斯平滑，使用上一步驟算出的  $\sigma_{\text{diff},i}$ ，產生出  $\text{num\_intervals}+3$  張模糊影像。

- 下採樣進入下一八度

完成當前八度所有尺度後，取該八度中「倒數第三張」(或類似規則) 的高斯圖，再做 1/2 下採樣，作為下一個八度的初始影像，重複相同程序。



Figure 6. 高斯金字塔

### 三、差分金字塔 (DoG) 與極值檢測

#### 1. 生成差分金字塔 (Generate DoG Images)

在一個八度內，相鄰兩張高斯模糊影像作差，就能得到差分金字塔的影格 (DoG layer)：

$$D(x, y, \sigma_i) = G(x, y, \sigma_{i+1}) - G(x, y, \sigma_i)$$

如此能在 DoG 影像中尋找局部最大/最小值，以取代拉普拉斯 (LoG) 的方法，節省運算成本。

#### 2. 尋找尺度空間極值 (Find Scale-Space Extrema)

在每個八度的 DoG 影像中，對第  $i$  層 ( $i$  介於 1 和  $num\_intervals$ )：

1. 取該層的 3D 立體鄰域：包含「前一層」( $\sigma_{i-1}$ )、「當前層」( $\sigma_i$ )、「後一層」( $\sigma_{i+1}$ ) 三張 DoG 影像的同位置  $(x, y)$  及其  $3 \times 3$  區域。
2. 若該點同時比鄰域所有像素都大，或都小，則為極值候選點。
3. 一併做初步的對比度閾值檢查（如對比度小於某門檻，直接忽略）。

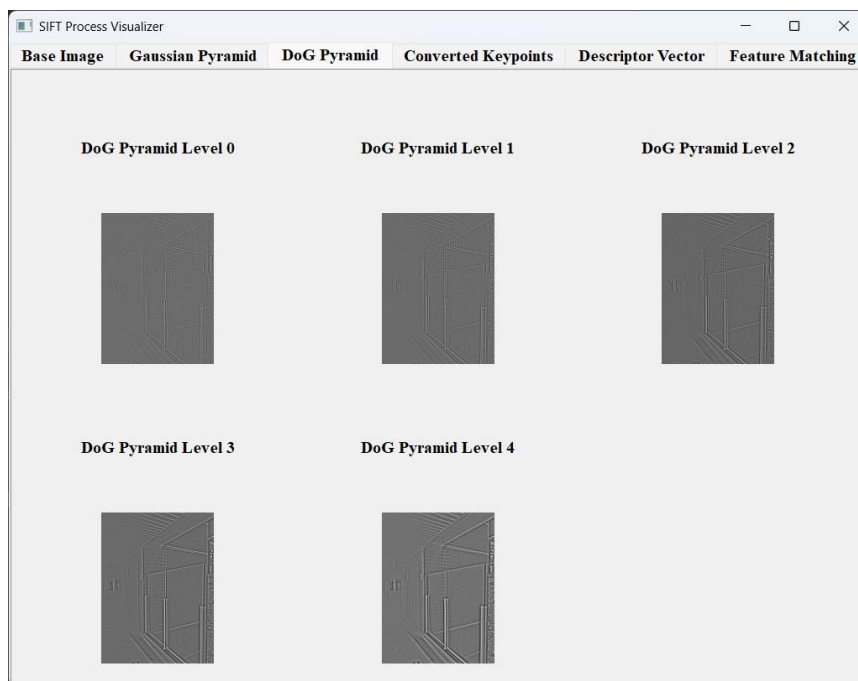


Figure 7. 高斯差分金字塔

### 四、亞像素精細化 (Quadratic Fit)

對於通過初步篩選的極值，SIFT 使用二次泰勒展開 (Taylor Expansion) 近似，來進一步在  $(x, y, \sigma)$  三維空間中求取更精準的位置。

主要包含：

#### 1. 梯度向量與 Hessian 矩陣

在當前影像立方體（前、中、後三層 DoG）中，對中心像素計算一階導數 (gradient) 與二階導數 (Hessian)。

### - 一階導數

$$\frac{\partial D}{\partial x}, \quad \frac{\partial D}{\partial y}, \quad \frac{\partial D}{\partial \sigma}$$

### - 二階導數 (Hessian)

$$\begin{bmatrix} \frac{\partial^2 D}{\partial x^2} & \frac{\partial^2 D}{\partial x \partial y} & \frac{\partial^2 D}{\partial x \partial \sigma} \\ \frac{\partial^2 D}{\partial y \partial x} & \frac{\partial^2 D}{\partial y^2} & \frac{\partial^2 D}{\partial y \partial \sigma} \\ \frac{\partial^2 D}{\partial \sigma \partial x} & \frac{\partial^2 D}{\partial \sigma \partial y} & \frac{\partial^2 D}{\partial \sigma^2} \end{bmatrix}$$

## 2. 極值位置微調 (Offset 計算)

以二次展開式在中心像素處的近似，可得極值在局部的偏移量：

$$\Delta = -H^{-1} \nabla D$$

其中  $\nabla D$  是一階導數向量， $H$  是 Hessian 矩陣。

解出  $\Delta$  即可得更精細的  $(\Delta x, \Delta y, \Delta \sigma)$ ，並更新該點的位置與尺度。

### - 對比度檢查 (Contrast Threshold)

更新後的 DoG 值若仍小於設定的對比度門檻，就捨棄該點。

### - 主曲率檢查 (Edge Response Elimination)

若 Hessian 矩陣中， $\text{trace}(H_{2 \times 2})^2 / \det(H_{2 \times 2})$  太大，代表該點更像是「邊緣響應」而非角點或穩定特徵，應排除。常用判斷：

$$\frac{(\text{trace}(H_{2 \times 2}))^2}{\det(H_{2 \times 2})} < \frac{(r+1)^2}{r}$$

其中  $r$  為可接受的主曲率比值上限。

## 五、方向分配 (Orientation Assignment)

對每個通過篩選的關鍵點，SIFT 會在其週圍區域計算梯度的方向直方圖，並找出最主要的方向作為關鍵點的「主方向」。若有其他「接近主要峰值 (80% 以上)」的峰，也會額外生成一個具有不同方向的關鍵點，以增加匹配時的穩定度。

### 1. 搜尋範圍與加權

在該關鍵點（所在八度空間的座標）附近，取一定半徑（與關鍵點尺度相關），對每個像素：

計算梯度幅度  $m$  與方向  $\theta$ 。

以高斯函式  $\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$  加權，令鄰域越遠的點權重越小。

### 2. 36-bin 方向直方圖

將  $\theta$  分配到對應的  $0^\circ \sim 360^\circ$  切成 36 個區間（每 bin 10 度）。

找出直方圖峰值  $\theta_{max}$

### 3. 多方向分配

取主峰值為第一方向。若有其他局部峰大於主峰的 80%，則也產生一個關鍵點（同座標與尺度，但不同方向）。

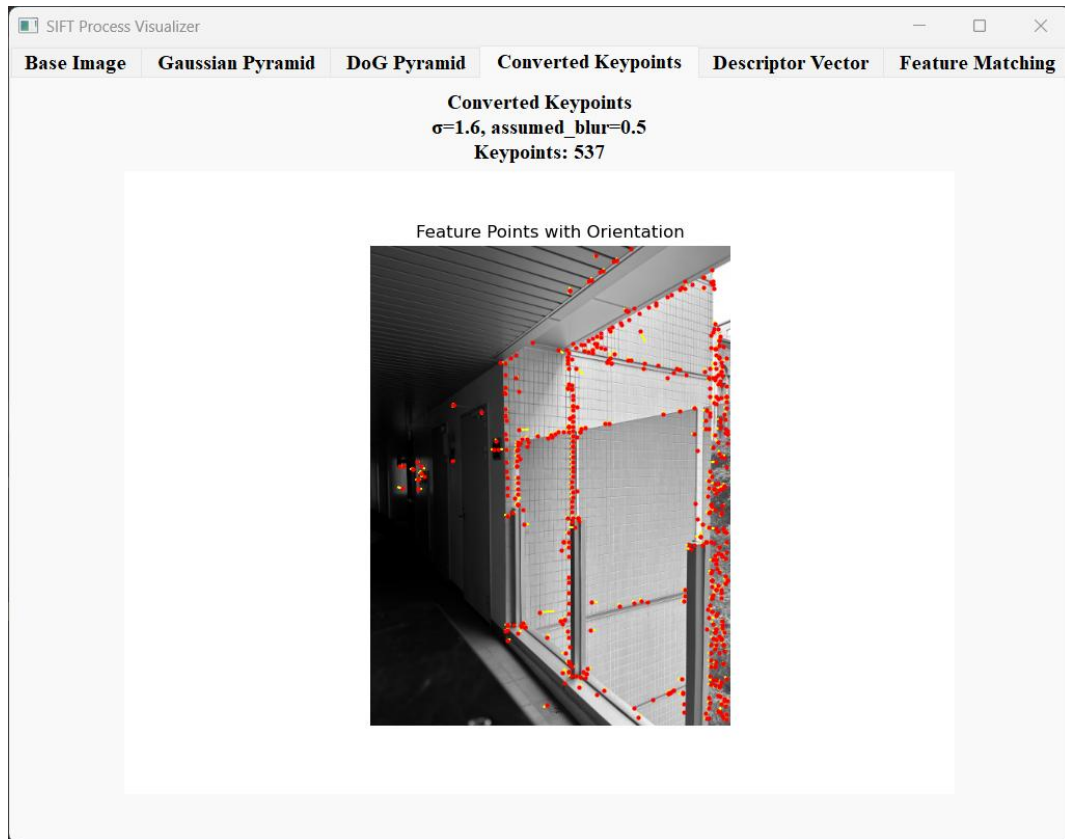


Figure 8. 含方向特徵點影像

## 六、重複鍵點移除與座標轉換

### 1. 刪除重複 (Remove Duplicate Keypoints)

在同一位置、同樣尺度、同樣方向上，可能產生多個重複關鍵點，故先做排序判斷，移除重複。

### 2. 轉回原圖大小 (Convert Keypoints to Input Image Size)

由於一開始將影像放大 2 倍，最終需要將關鍵點的坐標與 size 都縮回原圖比例（乘 0.5），以符合實際影像的位置。

## 七、描述子 (Descriptor) 生成

SIFT 描述子計算的核心，是在關鍵點附近取一個旋轉對齊的區域，分成  $4 \times 4$  子區域，每個子區域計 8 個方向直方圖，共 128 維 ( $4 \times 4 \times 8$ ) 向量。

### 1. 定位關鍵點中心與旋轉對齊

在該關鍵點（於所在 octave 的座標）周圍，取一個對應尺度大小的窗口，並依其「主方向」旋轉，使該窗口內的梯度方向相對於主方向對齊。

窗口大小與 `keypoint.size` 成正比，常乘以一條數（例如 3~4 倍）來決定描述子視窗半徑。



## 2. 計算梯度 (magnitude & orientation)

在該旋轉對齊的局部區域中：

每個像素做水平/垂直差分，得出  $(g_x, g_y)$ 。

幅度： $\text{mag} = \sqrt{g_x^2 + g_y^2}$ ，方向： $\theta = \arctan 2(g_y, g_x)$ 。

再以高斯函式對距中心越遠的像素進行權重衰減。

## 3. 將局部區域切成 4x4 格 (Spatial Binning)

令每個小格 (cell) 建立一個 8 維方向直方圖 (8 bins)。

當一個像素的方向角度落在某兩個相鄰 bin 之間，也會做線性或三線性插值分配 (通常還包含 row、col 的線性插值)，使描述子在空間、方向上都更平滑。

## 4. 描述子向量正規化與截斷 (Normalize & Clamp)

1. 先將整個 128 維向量做  $\ell_2$  正規化，使向量總長度為 1。
2. 將任何成分大於 0.2 (或類似設定) 者截為 0.2，以避免某些過大響應壟斷。
3. 再度正規化一次。
4. 最後將向量縮放至 0~255 的 byte 範圍，SIFT 原始實作常用  $\times 512$  再四捨五入到  $[0, 255]$ 。

最終得到的 128 維浮點或 byte 特徵向量，即為該鍵點的特徵描述子，可以用於後續特徵匹配。

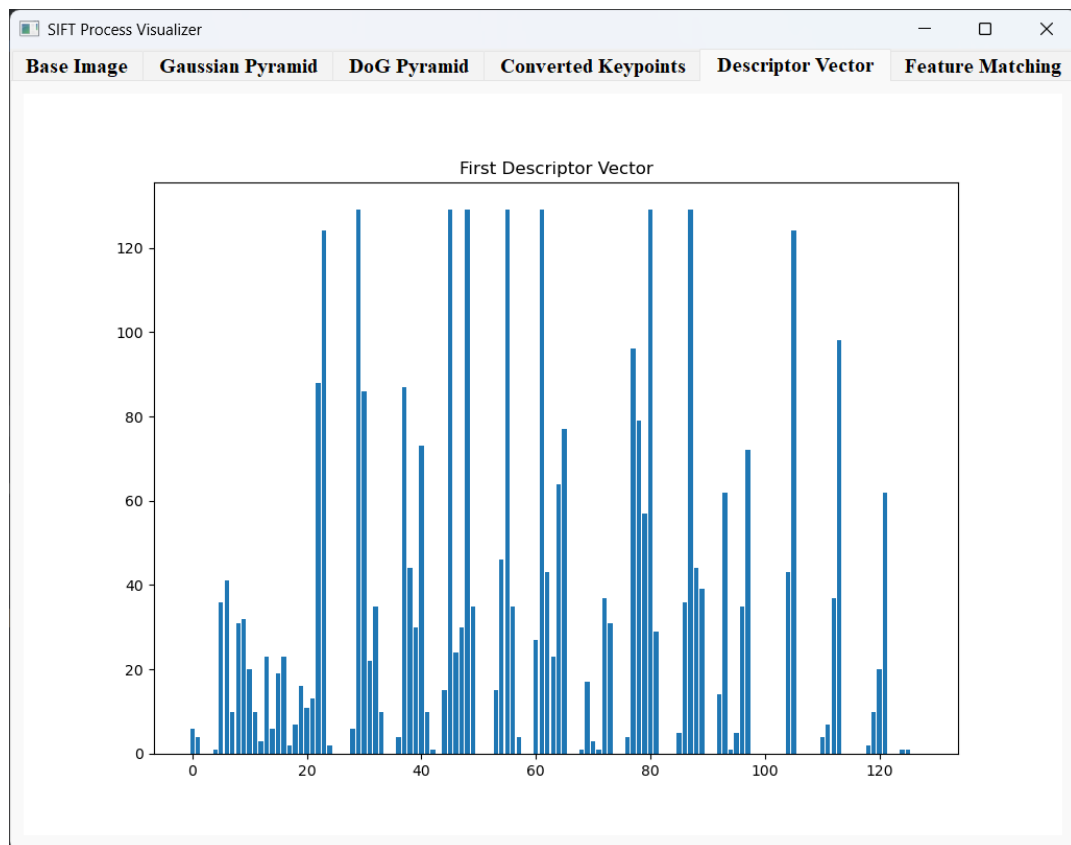


Figure 9. 128 維特徵描述向量分布

## 八、總結

這些步驟相互結合，構成了 SIFT 特徵點偵測與描述的完整流程，也正是程式中所實作的核心原理。透過對差分金字塔搜尋極值、二次泰勒展開做精細化、並對周圍鄰域梯度分佈計算描述子，SIFT 能達到對尺度、旋轉皆具備高度不變性的局部特徵偵測與描述。

### 2.2.2 Harris Corner Detector

以下將說明程式中所用到的 Harris 特徵偵測與特徵匹配（包含簡易描述子與 RANSAC）之流程，並剖析每個步驟背後的數學原理。

#### 一、Harris Corner Detector

Harris Corner Detector 用於找出影像中的角點 (corner)。角點通常被視為在兩條邊界交會處周圍，灰階或顏色分佈方向多變且對小範圍平移具敏感性的區域。計算流程

1. 灰階化：將彩色影像轉成灰階影像  $I_{\text{gray}}$ 。

2. 求梯度  $I_x, I_y$ ：使用簡單的離散微分（如 Sobel、或程式中自定義的 kernel  $H_x, H_y$ ）計算水平方向梯度  $I_x$  與垂直方向梯度  $I_y$ ：

$$I_x = \frac{\partial I_{\text{gray}}}{\partial x}, \quad I_y = \frac{\partial I_{\text{gray}}}{\partial y}$$

3. 累積矩陣 (Structure Tensor)：Harris 演算法中，會在局部區域內累積以下量：

$$A = I_x^2, \quad B = I_y^2, \quad C = I_x I_y$$

然後對這三張圖 (A, B, C) 做高斯平滑（或以 box filter 近似），得到平滑後的

$$\bar{A} = G_\sigma * (I_x^2), \quad \bar{B} = G_\sigma * (I_y^2), \quad \bar{C} = G_\sigma * (I_x I_y).$$

在某個像素 (x,y) 位置，可視為 2x2 矩陣

$$M = \begin{bmatrix} \bar{A}(x,y) & \bar{C}(x,y) \\ \bar{C}(x,y) & \bar{B}(x,y) \end{bmatrix}$$

#### 4. 角點響應函式 $R$

Harris 的角點判斷使用以下響應函式：

$$R = \det(M) - k \cdot (\text{trace}(M))^2,$$

其中

$$\det(M) = \bar{A}\bar{B} - \bar{C}^2, \quad \text{trace}(M) = \bar{A} + \bar{B}, \quad k \text{ 為經驗參數 } (0.04 \sim 0.06)$$

$\det(M)$  大代表兩個方向上的變化量都大，可能是角點。

$\text{trace}(M)$  大但  $\det(M)$  不大時，代表只是一條邊界（邊緣響應），非角點。

## 5. 閾值與區域最大值

設定一個閾值  $\text{threshold} = \alpha \times \max(R)$ ，若  $R(x, y)$  小於此閾值則忽略。

通過閾值者，再確認是否為其  $3 \times 3$  或更大鄰域中的最大值，若是則視為候選角點。

## 6. 取前 $\text{max\_points}$ 個角點

最後可依照  $R$  值從大到小排序，最多只取前幾個角點，避免角點過多，並確保保留高響應者。

程式最終會輸出一串角點清單，每個角點包含其座標  $(y, x)$  及對應的 Harris 響應值  $R$ 。同時也保留了梯度資訊  $I_x, I_y$  供後續描述子計算。

### 二、描述子 (Descriptor) 計算

程式採用類似 SIFT 的  $16 \times 16$  區域方向直方圖，不過是簡化版本，主要步驟如下：

#### 1. 取得角點附近 $16 \times 16$ 區域 (patch)

對於每個角點  $(y, x)$ ，為了取出一個局部區域以供特徵量化，需要確保周圍  $16 \times 16$  區域在影像範圍內，因此常設置一個邊界保護 ( $\text{margin} = 8$ )。

#### 2. 計算梯度大小 $m$ 與方向 $\theta$

先在整張影像（或每個 patch）中，利用上一步的

$$m = \sqrt{I_x^2 + I_y^2}, \quad \theta = \arctan 2(I_y, I_x) \text{ (以度數表達且範圍 } 0^\circ \sim 360^\circ \text{)}.$$

然後在  $16 \times 16$  patch 中取對應的  $m, \theta$ 。

#### 3. 找主要方向 (main orientation)

先對  $16 \times 16$  區域整體的梯度方向做一個粗略統計，可建立 8 個 bin (將  $0 \sim 360$  度分成 8 區，每 45 度一個 bin)，並將該 patch 所有像素的梯度大小累加到對應 bin。bin 數值最大的角度區間即該區塊「主要方向」，記作  $\theta_{\text{main}}$ 。

#### 4. 方向對齊 (Orientation Alignment)

之後會將整個 patch 的角度值視為「 $\theta - \theta_{\text{main}}$ 」，使得該角點的描述子對此主要方向做旋轉對齊，亦即保有旋轉不變性。

#### 5. 分成 $4 \times 4$ 子區域、計算方向直方圖

將這個  $16 \times 16$  區域再分割成  $4 \times 4$  個小塊，每小塊大小為  $4 \times 4$ 。在每個小塊中，再次以 8-bin 的方式統計方向分佈，不過這次是針對「旋轉對齊後的角度」做統計。最後可得到  $4 \times 4 \times 8 = 128$  維的向量，作為該角點描述子。

#### 6. 描述子正規化

SIFT 風格會對整個 128 維向量先做  $l_2$  正規化，使向量總長度為 1。若有成分大於 0.2（或其他閾值），則將其截斷為 0.2，之後再做一次  $l_2$  正規化，藉此抑制過大響應對特徵造成不平衡。

最終得到每個角點對應的 128 維描述子，與  $(x, y)$  座標一一對應。

### 三、總結

透過以上流程，可以找到兩張影像之間的大致位移關係，並在後續步驟中將影像做平移、疊合與拼接。程式也在此基礎上進一步執行圓柱投影 (cylindrical projection)、影像融合 (blend) 及裁切 (crop) 等操作，最終得到全景拼接效果。

## 2.3 Feature Matching

這兩個程式分別使用了 **SIFT** 與 **Harris + 簡易描述子** 兩種不同的特徵偵測與描述子生成方式，但在匹配與 RANSAC 部分的原理大致相同。以下依序說明：

### 1. 最近鄰匹配 (Nearest Neighbor Matching)

無論是 SIFT 還是 Harris+簡易描述子，最終都得到一組「特徵向量 (128 維)」，在程式中，透過以下簡化的最近鄰匹配：

- 針對影像 A 的每個特徵向量  $\mathbf{d}_A$ ，在影像 B 特徵集中找出使 L2 距離最小的  $\mathbf{d}_B$ 。
- 若該最小距離小於一個設定的閾值 (程式中如 `desc_thresh=25000` 或 `1.0` 等，視描述子大小而定)，則接受此匹配。
- 形成匹配對  $((x_A, y_A), (x_B, y_B))$ 。

距離的計算公式 (以 L2 為例)：

$$\text{dist}(\mathbf{d}_A, \mathbf{d}_B) = |\mathbf{d}_A - \mathbf{d}_B|^2 = \sum_{k=1}^{128} (dA_k - dB_k)^2$$

若該值 < 閾值，表示描述子非常相似，可能是真正匹配。

### 2. RANSAC 找平移 (dx, dy)

因為此拼接僅考慮純平移 (或已經透過圓柱投影消弭了大部分透視變化)，因此對於所有匹配對  $((x_A, y_A), (x_B, y_B))$ ，可計算

$$\Delta x = x_A - x_B, \quad \Delta y = y_A - y_B.$$

若同一對影像真實的平移量為  $(dx^*, dy^*)$ ，那麼越多匹配都應落在此平移量附近。程式中做法如下：對所有匹配對計算  $(\Delta x, \Delta y)$ 。

任取一個  $(\Delta x_0, \Delta y_0)$  當作假設：

計算其他匹配對與此假設的差距  $((\Delta x_i - \Delta x_0)^2 + (\Delta y_i - \Delta y_0)^2)$ ，若小於設定閾值 (如 33)，則計為 inlier。

逐一嘗試，最後取 inlier 數量最多者為最終平移量。

此即 **RANSAC** (隨機採樣一致性) 的簡化版本，用投票 (voting) 方式找最大公約數之移動向量。



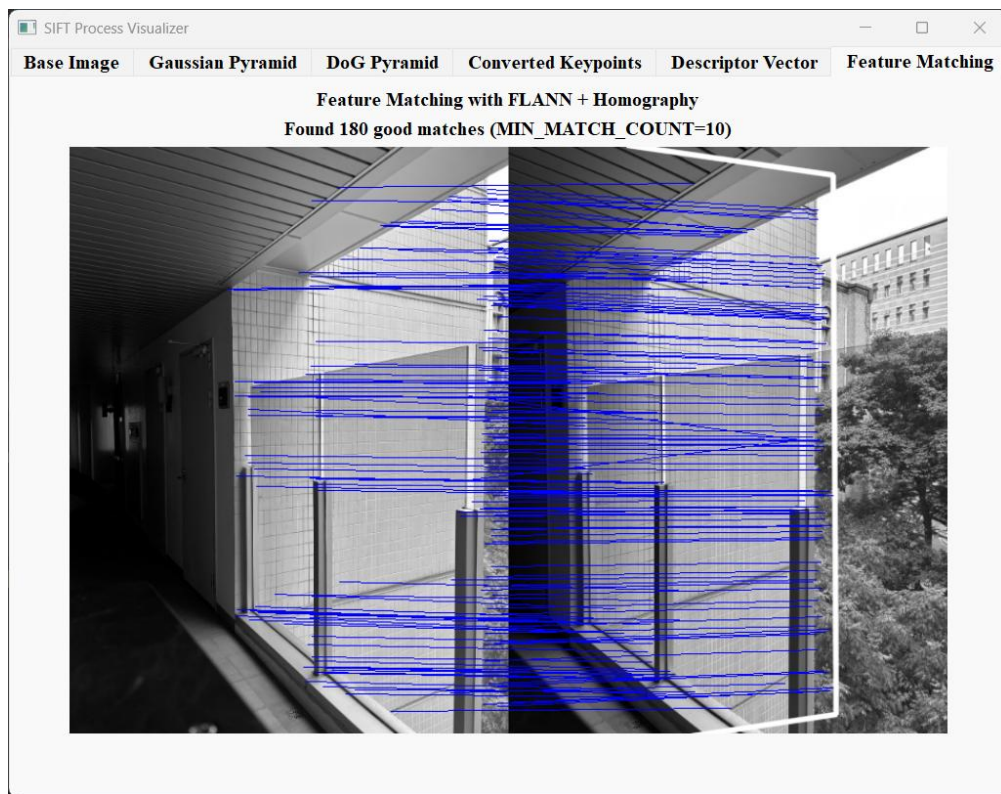


Figure 10. SIFT 特徵匹配示意圖(此視覺化模型有配合 opencv 函式庫，主程式沒有使用)

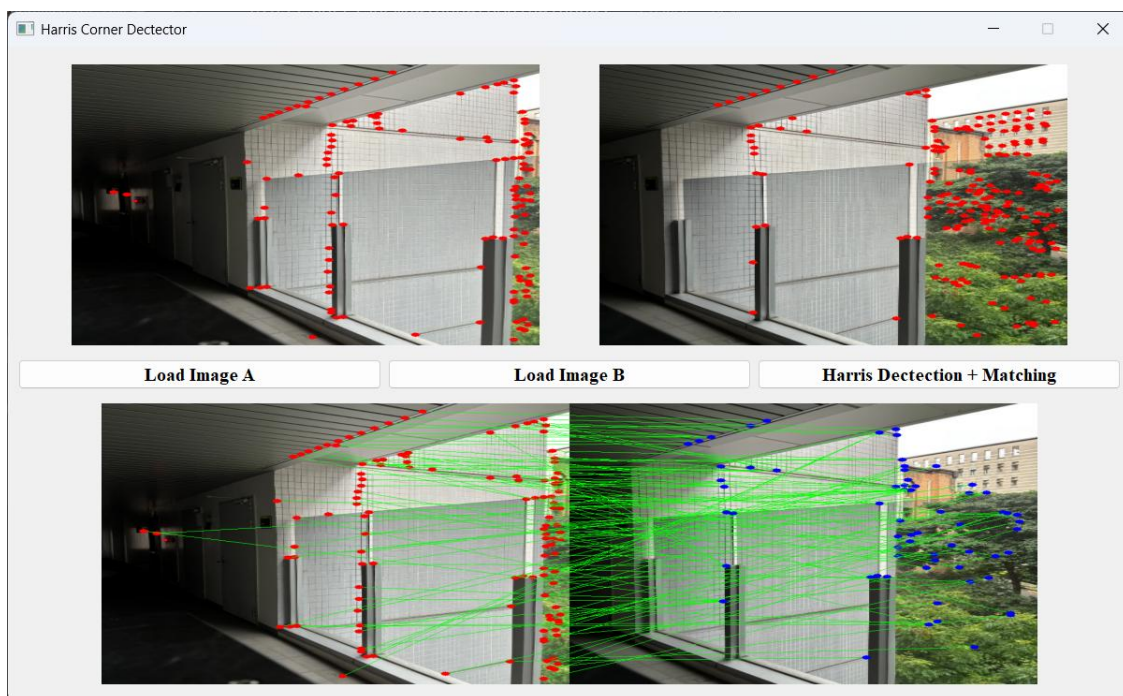


Figure 11. Harris 特徵匹配示意圖

## 2.4 Image Matching

所謂「影像匹配」在此主要指的是根據前一步算出的  $(dx, dy)$  來對新影像做適當的「平移」，然後把它「接」到前面已拼好的全景上。由於這些程式都用了「圓柱投影」(cylindrical projection) 來處理相機擺動對水平方向的扭曲，故最終只需考慮水平/垂直的簡單平移。

### 1. 圓柱投影 (Cylindrical Projection)

假設焦距 =  $f$ ，對於影像中某像素  $(x, y)$ ，令影像中心在  $(c_x, c_y)$ 。程式中計算方式：  
先得圓柱投影

$$x_{\text{dist}} = x - c_x, \quad y_{\text{dist}} = y - c_y.$$

$$x_{\text{mapped}} = \left\lfloor f \cdot \arctan\left(\frac{x_{\text{dist}}}{f}\right) \right\rfloor + c_x, \quad y_{\text{mapped}} = \left\lfloor f \cdot \frac{y_{\text{dist}}}{\sqrt{x_{\text{dist}}^2 + f^2}} \right\rfloor + c_y.$$

其中  $\lfloor \cdot \rfloor$  表示四捨五入。

透過此步驟，可以將原本視角較廣的圖轉成類似「捲在圓柱上」的效果，以減少透視變形，讓影像可以用簡單平移便可「大致」對齊。



Figure 12. 範例為測試數據 /grail 拼接四張圖片後結果，可以看出明顯的圓柱投影形狀

### 2. 累計平移與 drift 修正

當要拼多張影像，程式會計算相鄰兩張之間的平移量  $(dx_i, dy_i)$ ，不斷把新影像「接」到已拼好的全景影像上。為防止因誤差累積而造成「第一張和最後一張」在高度上有漂移 (drift)，常會計算最後得到的整體  $(dx, dy)$ ，再將其等分到每一步的  $(dy)$  做修正，減少上下飄移。

流程概念：

1. 第 0 張當作基準，之後每張的累計移動量 = 前面累計 + 當前平移；
2. 最後  $\Delta y$  可能大於 0，表示影像越拼越往上或往下；
3. 在第二輪實際拼接時，將每一步的  $dy$  減去一個平均漂移量  $\Delta y \left( \frac{\text{張數} - 1}{\text{張數} - 1} \right)$ 。
4. 重新拼接可得到較平滑的水平對齊。

### 3. 影像平移 (padding)

程式中用 `pad_image` 函式，依  $(dx, dy)$  做 zero padding。例如要把一張影像 B 向右下平移  $(dx > 0, dy > 0)$ ，就上方與左方都 pad 多少行/列的 0，並將原圖貼到該區塊的後方。如此合成後，兩張影像就產生相對位移效果。



Figure 13. 未解決 drift 前，parrington 拼接結果



Figure 14. 解決 drift 後，parrington 拼接結果

## 2.5 Blending

完成了「影像對齊」之後，若兩張影像有重疊區域，程式就會使用簡單的線性漸變融合來避免硬疊造成的接縫 (seams)。以下為主要概念：

### 1. 重疊區 (Overlap)

程式計算兩張平移後的影像在  $x$  方向上重疊的範圍 (`overlap_range`)。之後在這個範圍內，設定一個隨著列 (`column`) 遞增的「alpha」，做線性混合。

### 2. 線性混合公式

令  $\alpha$  為 0 到 1 之間，對應於重疊區域的左至右。對於每個像素 (同一行  $c$ )，分別從兩張影像 ( $A, B$ ) 取得像素值  $p_A, p_B$ 。融合後結果可定義為：

$$p_{blend} = (1 - \alpha) p_A + \alpha p_B.$$

程式中的 `alpha` 是：

$$\alpha = \frac{\text{overlap\_counter}}{\text{overlap\_range}},$$

`overlap_counter` 每往右一列就加一，實現從 0 漸漸到 1 的過程，達到漸層轉換。

### 3. 非重疊區處理

若某列只有 A 有內容 ( $B$  為 0) 就直接用 A；只有 B 有內容就直接用 B；兩者皆無則維持 0；兩者都有則做上述線性混合。



#### 4. 最後裁切 (rectangle\_crop)





當整張拼接好的影像外圍仍有黑色無效區域，程式中會對灰階值做閾值判斷，找出有效區域的最小外框並裁切，以得到較乾淨的結果。各數據集裁切邊界大小建議： /grail = 17, /parrington = 15, /out = 30, /wind = 24



Figure 15. Rectangling 後，parrington 拼接結果


### 3. RESULTS

本研究使用以下四種數據做影像拼接：

Grail	Parrington	Lab Outside	Genshin Impact
			
18 張	18 張	8 張 (最後用兩張)	8 張 (最後只用兩張)

使用兩種演算法與 autostitch 分別做比較：

Grail =  
上: SIFT (執行時間 1174.90 秒)  
中: Harris (執行時間 22.12 秒)  
下: Autostitch (執行時間 10 秒左右)





Parrington =

上: SIFT (執行時間 1446.45 秒)

中: Harris (執行時間 19.38 秒)

下: Autostitch (執行時間 10 秒左右)



Lab Outside =

左: SIFT (執行時間 1174.90 秒)

中: Harris (執行時間 2.12 秒)

右: Autostitch (執行時間 3 秒左右)



Wind =

上: SIFT (執行時間 0.78 秒) 匹配不正確, Timer: 0.00 秒 SIFT 運算

中: Harris (執行時間 0.93 秒) 匹配不正確, Timer: 0.00 秒 Harris 角點 + RANSAC 完成

下: Autostitch (執行時間 2 秒左右)



新增程式功能比較：



## 4. CONCLUSIONS

### Feature Detection and Matching

先以 SIFT 或 Harris+描述子找到兩張影像的特徵點與特徵向量；

用最近鄰 (L2 距離) 進行初步匹配；

用 RANSAC 選出最佳平移量，排除離群值。

### Image Matching

由 RANSAC 得到的  $(dx, dy)$  表示新影像相對舊影像的平移；

配合圓柱投影 (cylindrical projection) 消除鏡頭旋轉帶來的曲面扭曲；

累計多張影像的平移量，並可根據「首尾之差」做 drift 修正，保證不會越拼越高/低。

### Blending

在重疊區域用線性漸變混合，使影像邊緣平滑過渡；

若有多列或多行重疊也可類似做二維的權重漸變。

拼完後可能周圍仍有黑邊，最後做裁切。



透過以上關鍵演算法與數學公式的結合，就能在實作中完成多張影像的全景拼接：

1. 特徵偵測 (SIFT / Harris)
2. 特徵與影像匹配 (Nearest Neighbor + RANSAC)
3. 影像對齊 (平移)
4. 影像融合 (簡單線性 blending)
5. 裁切 (rectangle\_crop)

#### Data Availability Statement:

All the codes, datasets and files can be viewed and downloaded via the links : [Github : Image Stitching](#)

## REFERENCES

- [1] Richard Szeliski and Heung-Yeung Shum. 1997. **Creating full view panoramic image mosaics and environment maps**. In Proceedings of the 24th annual conference on Computer graphics and interactive techniques (SIGGRAPH '97). ACM Press/Addison-Wesley Publishing Co., USA, 251–258. <https://doi.org/10.1145/258734.258861>
- [2] Harris, Christopher G. and M. J. Stephens. “A Combined Corner and Edge Detector.” *Alvey Vision Conference* (1988). <https://doi.org/10.5244/C.2.23>
- [3] Brown and Lowe, “**Recognising panoramas**,” Proceedings Ninth IEEE International Conference on Computer Vision, Nice, France, 2003, pp. 1218-1225 vol.2, <https://doi.org/10.1109/ICCV.2003.1238630>.
- [4] Lowe, D.G. **Distinctive Image Features from Scale-Invariant Keypoints**. *International Journal of Computer Vision* **60**, 91–110 (2004). <https://doi.org/10.1023/B:VISI.0000029664.99615.94>
- [5] Brown, M., Lowe, D.G. **Automatic Panoramic Image Stitching using Invariant Features**. *Int J Comput Vision* **74**, 59–73 (2007). <https://doi.org/10.1007/s11263-006-0002-3>
- [6] [Implementing SIFT in Python: A Complete Guide](#)