

NTU-VFX-Project2 Report

Team Member

- R10922022 曾筱晴
- B09902028 曾翊綺

Description

In this project, we have to implement image stitching, i.e., combine a set of images to create the panoramas.

Implementation

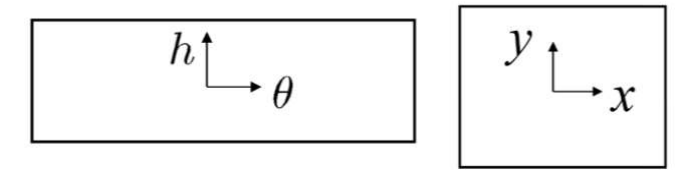
- **Image Warping**
 - Cylindrical Warping
- **Feature Detection**
 - Harris Corner Detector
- **Feature Description**
 - SIFT Descriptor
- **Feature Matching**
 - Brute-force Search
- **Image Matching**
 - RANSAC
- **End-to-End Alignment**
 - Global Wrap
- **Image Blending**
 - Linear Blending

Step 0. Focal Length

- 透過 Autostitch 獲得每張影像的 focal length。（在 pano.txt 中）

Step 1. Image Warping

- 使用 Cylindrical Warping 的方式將影像投影到圓柱上。（以下的 x 、 y 是以影像中心為原點的座標系統）







- 利用以下公式去計算每個 pixel 的 θ 及 h 。

$$\theta = \tan^{-1} \frac{x}{f}, \quad h = \frac{y}{\sqrt{x^2 + f^2}}$$

- 將 θ 及 h 乘上 scale s 作為新的 x 、 y 座標，而這邊使用 focal length 作為 s （較少 distortion）。

$$x' = s\theta = s \tan^{-1} \frac{x}{f}, \quad y' = sh = s \frac{y}{\sqrt{x^2 + f^2}}$$

- 最後，將左右兩側的黑色區域裁減掉，方便後續的拼接。

Original Image		Cylindrical Warping	
			
No Cylindrical Warping			
Cylindrical Warping			

Step 2. Feature Detection

實作 Harris Corner Detector





- 先算出影像中 x, y 方向上的 derivative : I_x, I_y 。
- 計算 I_{x^2}, I_{y^2}, I_{xy} 。
- Gaussian Blur 得到 S_{x^2}, S_{y^2}, S_{xy} , 即可得到 matrix M 。

$$M(x,y) = \begin{bmatrix} S_{x^2}(x,y) & S_{xy}(x,y) \\ S_{xy}(x,y) & S_{y^2}(x,y) \end{bmatrix}$$

- 算出每個 pixel 的 R 值。

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

- threshold R, 把 R 值太小的設為 0。
- 對 threshold 完的 R 取 local maximum。

Original Image	Corner Response R
	
Local Maximum on R	Key Points
	

Example

	
---	---

Step 3. Feature Description

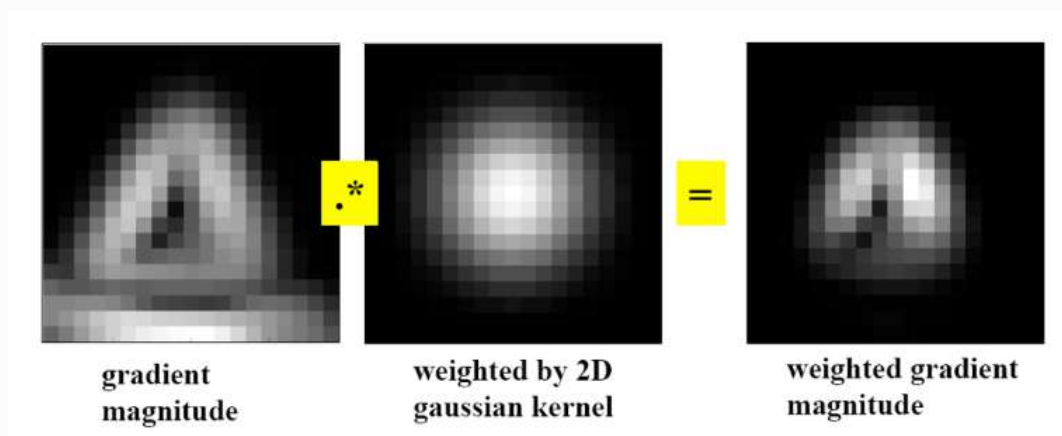
- 利用以下公式計算整張 image 每個 pixel 的 gradient 大小以及方向。

L : Gaussian-smoothed image with the closest scale.

$$m(x, y) = \sqrt{(L(x+1, y) - L(x-1, y))^2 + (L(x, y+1) - L(x, y-1))^2}$$

$$\theta(x, y) = \tan^{-1}((L(x, y+1) - L(x, y-1)) / (L(x+1, y) - L(x-1, y)))$$

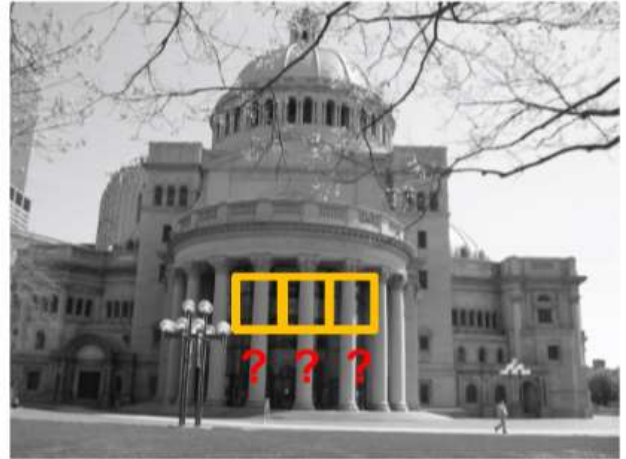
- 對所有 key point 附近的 32×32 區塊去做旋轉，旋轉後取中間的 16×16 作為 Local Patch。
- 將 Local Patch 分成 16 塊，每塊 4×4 ，而每個小塊裡的每個 pixel 對 8 個方向去做投票，即可得到一個 histogram。（詳細作法如下）
- 將 360 度分成 8 個 bins，查看 gradient 的方向屬於哪個 bin，使用 gradient magnitude 做完 gaussian 後的大小作為該 pixel 能投的票數（如下圖所示）。



- 將 histogram 做 normalize，並將大於 0.2 的值都設為 0.2，接著再做一次 normalize，即為該小區塊的 descriptor。
- 而每個小區塊的 descriptor 合併在一起後，即為最終的 SIFT descriptor。（一個 Local Patch 分成 4×4 塊，每小塊 8×1 的 histogram，共 128 維）

Step 4. Feature Matching

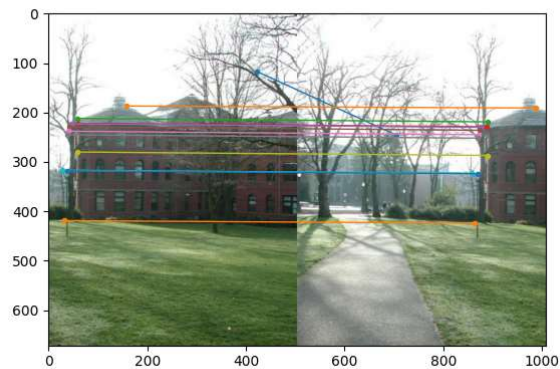
- 使用 scipy package 中的 `cdist`，計算兩張影像每個 descriptor 之間的距離。
- 為了解決 ambiguous matches 的問題，找出和每個 descriptor 最接近的和第二接近的 descriptor（distance 最小和第二小的）。若兩者之間的 ratio 接近 1，則表示兩個之間的差異很小，很有可能是 ambiguous match，因此捨棄該 match。



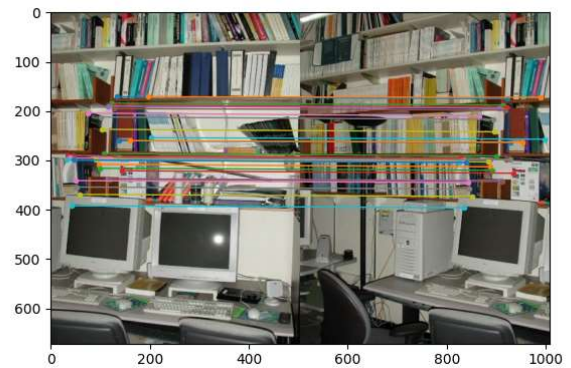
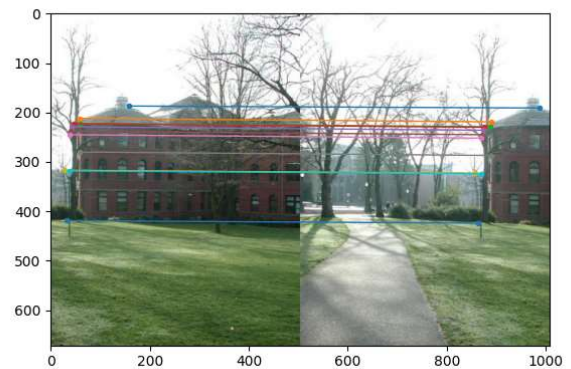
Step 5. Image Matching

- 使用 RANSAC，將先前找到的 matches 中的 outlier 剔除。
- 做多次的 iteration，每次隨機挑 5 組的配對點，計算那五組配對點 x 方向的平均位移（因為使用腳架拍攝 y 方向的變化較不明顯，所以這邊使用 x 方向的位移作為判斷條件）。
- 找出與這五組配對點 x 方向平均位移差距 $\leq n$ 個 pixel 的 match (inlier)，若 inlier 的數量比先前找到的多，則記下這些 inlier。
- （因為每張影像適用的 n 不同，所以這邊使用動態調整，若 iteration 結束後找到的 inlier 小於所有 match 數量的 $1/3$ ，則增加 n 的大小重新迭代。）
- 最後，利用所有的 inlier 去計算 x 、 y 方向的 translation。

Before RANSAC

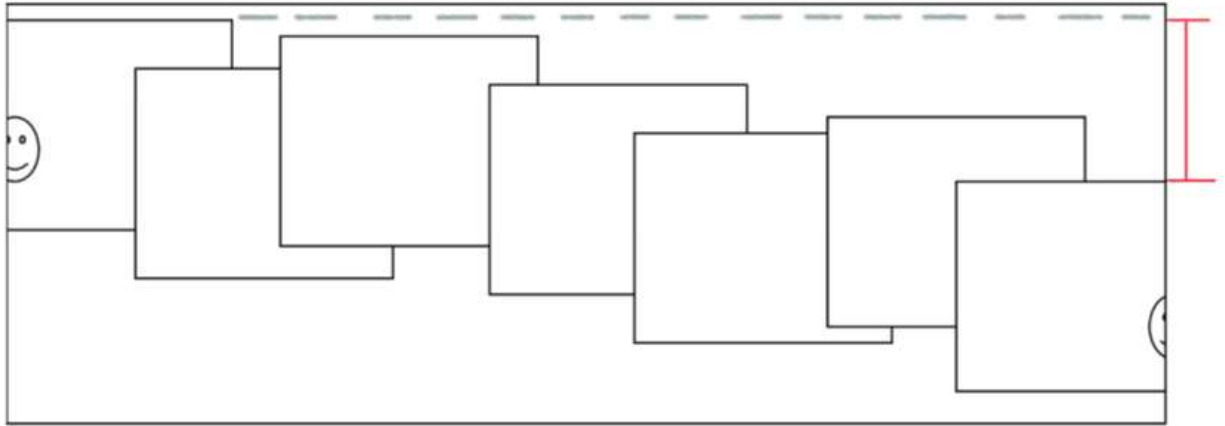


After RANSAC



Step 6. End-to-End Alignment

- 計算第一張和最後一張影像之間的 y 方向的 drift。



- 將該 drift 平均分配到倆倆的影像之間。

$$y' = y + dy, dy = \frac{y_{drift}}{(n-1)}, n \text{ is number of images}$$

Non-Alignment	
Alignment	

Step 7. Image Blending

- 使用 linear blending
- 不是對於所有 overlap 的區域都做 linear blending，只取邊界內的特定範圍（code 中的 `percent` 變數）
- 取 overlap 的中線作為兩張圖片的交界，從中線往右/左擴張，右/左圖片的權重會逐漸增加，到最後就會是完全取右/左的 pixel 值了

Results

parrington



grail



stairs



road



心得

以前看到別人找特徵點的影像都覺得很酷，到底是怎麼算才能這麼精確，全景的部分平常很常用手機拍，不過都沒想過有天自己也能接觸、了解背後的演算法，並自己實作，很有成就感。

不過，實作的過程中有很多小細節需要注意，參數需要依照不同的影像狀況來做調整，例如：在跑 RANSAC 的時候，threshold 的設定會因為所拍攝的影像解析度不同、遠近的物體移動距離的不同等原因而有所差異，一旦沒有設定好結果就會很不穩定。